UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Mitja Mihelič

# Migracija podatkov iz podatkovne zbirke MySQL na Percona Server in MariaDB

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

Ljubljana, 2014

UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Mitja Mihelič

# Migracija podatkov iz podatkovne zbirke MySQL na Percona Server in MariaDB

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

MENTOR: viš. pred. dr. Aljaž Zrnec

Ljubljana, 2014

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Pred leti je Oracle prevzel vodenje ene izmed najbolj popularnih odprtokodnih implementacij relacijske podatkovne baze MySQL. Zaradi statusa in samega poteka razvoja podatkovne baze MySQL pod okriljem podjetja Oracle, se je ustanovitelj MySQL-a, Michael "Monty" Widenius odločil, da razvije novo podatkovno zbirko MiariaDB. Iz enakega razloga je nastal tudi produkt Percona Server. Na prej omenjeno podatkovno bazo MariaDB bo iz MySQL-a migriral tudi eden izmed velikanov v današnjem informacijskem svetu, Google. V okviru diplomskega dela ugotovite, kako se izvede migracija podatkov iz podatkovne baze MySQL na podatkovni bazi MariaDB in Percona Server. Pri tem se osredotočite predvsem na posebnosti na katere je potrebno biti pozoren pri migraciji, kaj je treba popraviti v že narejenih aplikacijah, ki uporabljajo MySQL, s katerimi pastmi se bodo razvijalci srečevali in kakšna naj bi bila nemotena ter pravilno izvedena migracija podatkov.

# IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Spodaj podpisani Mitja Mihelič, z vpisno številko **63980093**, sem avtor diplomskega dela z naslovom:

*Migracija podatkov iz podatkovne zbirke MySQL na Percona Server in MariaDB*

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom viš. pred. dr. Aljaža Zrneca
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 26. septembra 2014 Podpis avtorja:

*Zahvalil bi se rad mojemu mentorju viš. pred. dr. Aljažu Zrnecu, ki me je pravilno usmerjal pri izdelavi diplomske naloge in mi dajal koristne in uporabne nasvete. Zahvalil bi se tudi očetu, ki me je vsa dolga leta vzpodbujal h končanju študija. Posebno pa bi se rad zahvalil še moji družini, soprogi Brini, hčerki Algaji in sinu Tristanu, ki so mi dajali podporo in me polnili s pozitivno energijo.*

*Svojima dragima otrokoma, hčerki Algaji in sinu Tristanu.*

# Kazalo

**Povzetek**

**Abstract**

**1 Uvod ..... 1**

**2 Predstavitev podatkovnih zbirk ..... 3**

2.1 MySQL ..... 3

2.2 MariaDB ..... 5

2.3 Percona Server ..... 7

**3 Primerjava podatkovnih baz ..... 9**

3.1 Primerjava lastnosti podatkovnih baz ..... 9

3.2 Podatkovni stroj InnoDB ..... 11

3.3 Podatkovni stroj XtraDB ..... 15

**4 Migracija podatkov ..... 17**

4.1 Potek migracije na MariaDB ..... 17

4.2 Potek migracije na Percona Server ..... 19

**5 Praktična izvedba migracije na podatkovno zbirko MariaDB ..... 21**

5.1 Uporabljena strojna in programska oprema ..... 21

5.2 Izdelava testne podatkovne baze in testne aplikacije ..... 22

5.3 Zaključek ..... 27

**6 Sklepne ugotovitve ..... 29**

**Literatura ..... 31**

**Spletni viri ..... 31**

**Priloge ..... 33**

- Priloga 1 ..... 33
- Priloga 2 ..... 38

- Priloga 3 ..... 39

# Seznam uporabljenih kratic

| kratica | angleško | slovensko |
|---|---|---|
| **ACID** | Atomicity, Consistency, Isolation, Durability | atomarnost, konsistentnost, izolacija, trajnost |
| **API** | Application Programming Interface | aplikacijski programski vmesnik |
| **APT** | Advanced Package Tool | program za upravljanje namestitev in odstranjevanje programov na operacijskih sistemih Linux |
| **CPU** | Central Processing Unit | centralno procesna enota |
| **DBMS** | Database Management System | sistem za upravljanje s podatkovno bazo |
| **DML** | Data Manipulation Language | jezik za vodenje podatkov |
| **EER** | Enhanced Entity Relationship | diagram razmerja med entitetami |
| **GPL** | General Public License | licenca za prosto programje |
| **GUI** | Graphical User Interface | grafični uporabniški vmesnik |
| **IO** | Input - Output | vhod - izhod |
| **LAMP** | Linux, Apache, MySQL, Perl/PHP | Linux, Apache, MySQL, Perl/PHP |
| **MSI** | Windows Installer | namestitvenik za OS Windows |
| **NoSQL** | Not only Structured Query Language | ne samo poizvedovalni jezik v relacijskih podatkovnih bazah |
| **NUMA** | Non-uniform memory access | neenotni dostop do pomnilnika |
| **RDBMS** | Relational Database Management System | sistem za upravljanje relacijskih podatkovnih baz |
| **RW** | Read - Write | branje - zapisovanje |
| **SQL** | Structured Query Language | poizvedovalni jezik v relacijskih podatkovnih bazah |

# Povzetek

Pred leti je Oracle prevzel vodenje ene izmed najbolj popularnih odprtokodnih implementacij relacijske podatkovne baze MySQL. Zaradi statusa in samega poteka razvoja podatkovne baze MySQL pod okriljem podjetja Oracle, se je ustanovitelj MySQL-a Michael "Monty" Widenius odločil, da razvije novo podatkovno zbirko MariaDB. Iz enakega razloga je nastal tudi produkt Percona Server, ki ga je mogoče namestiti samo na operacijskem sistemu Linux. Na prej omenjeno podatkovno bazo MariaDB bo iz MySQL-a migriral tudi eden izmed velikanov v današnjem informacijskem svetu, Google.

Diplomsko delo obravnava tematiko o izvedbi migracije podatkov iz podatkovne zbirke MySQL na podatkovno zbirko MariaDB in Percona Server. Pri tem se osredotoča predvsem na posebnosti na katere je potrebno biti pozoren pri migraciji podatkov, na enakosti in razlike med omenjenimi bazami, kaj je treba popraviti v že narejenih aplikacijah, ki uporabljajo MySQL, s katerimi pastmi se bodo razvijalci srečevali pri migraciji in kakšna naj bi bila nemotena ter pravilno izvedena migracija podatkov. Prikazan je tudi primer testne podatkovne baze in aplikacije, s pomočjo katere se izvede test delovanja aplikacije pred migracijo in po migraciji na novi podatkovni bazi MariaDB.

**Ključne besede:** podatkovne baze, MySQL, Percona Server, MariaDB, migracija podatkov

# Abstract

Few years ago, Oracle took over one of the most popular open source relational database management systems (DBMS) implementations, the MySQL. Due to the events that followed, the MySQL AB co-founder Michael "Monty" Widenius left the company, only to later develop a new open source database system, called Maria DB. An equivalent, Linux specific implementation, called Percona Server was also developed as a separate project but with same goals in mind. Together with numerous customers that have chosen the new MySQL supplement, one of the industries giants, Google, also chose the new MariaDB for its services.

This BSc Thesis aims to present the process of migration from MySQL to both MariaDB and Percona Server, highlighting best practice tips to navigate the migration process the easiest way on one hand, and the constraints and problems most likely encountered during this process on the other. It also presents most basic differences between these implementations. It also demonstrates the migration process from MySQL to MariaDB through a pre and post migration testing scenario of an operational database environment with test application.



**Keywords:** databases, MySQL, Percona Server, MariaDB, data migration

# 1 Uvod

Podatki, ki se shranjujejo in so shranjeni v podatkovnih bazah, so osnova za delovanje vseh sistemov. Zgodovina shranjevanja podatkov sega v zgodnja šestdeseta leta prejšnjega stoletja, ko je Charles Bachman v podjetju General Electric razvil prvi DBMS, Integrated Data Store. Že v sedemsetih letih prejšnjega stoletja pa Edgar Codd iz podjetja IBM predlaga relacijski podatkovni model in tako se posledično v teh letih razvije mnogo relacijskih DBMS-jev in relacijske podatkovne baze postanejo standard za upravljanje s podatki v organizacijskih sistemih. V osemdesetih letih prejšnjega stoletja si relacijski podatkovni model samo še utrjuje položaj kot DBMS. V teh letih se tako razvije tudi poizvedovalni jezik SQL, ki postane standardni jezik za izvajanje poizvedb v relacijskih podatkovnih bazah.

Leta 1995 je švedsko podjetje MySQL AB, ketere ustanovitelji so bili David Axmark, Allan Larsson in Michael »Monty« Widenius, razvili relacijsko podatkovno bazo MySQL in jo poimenovali po Michaelovi hčerki My. Že od vsega začetka je bila podatkovna zbirka MySQL zelo priljubljena in je hitro postala del LAMP, skupka odprtokodne programske opreme, ki skupaj tvori popolnoma delujoč spletni strežnik in ki je sposobna gostiti dinamične spletne strani. Na začetku je bil MySQL brezplačen za osebno uporabo, za uporabo na komercialnih straneh in strežnikih Windows pa si moral kupiti komercialno licenco. V letu 2000 so brezplačno licenco spremenili v GPL, kar je pomenilo, da je produkt postal brezplačen tudi za komercialno uporabo. Leta 2006 se je za nakup produkta MySQL začelo zanimati podjetje Oracle. Lastniki MySQL-a so zavrnili ponudbo. Leta 2008 pa so se odločili, da jih prevzame podjetje Sun za milijardo dolarjev. Prevzem podjetja Sun je pomenil tudi prevzem talentov, ki so ustvarili produkt MySQL. Vendar sta Widenius in Axmark še isto leto zapustila podjetje in Widenius je celo javno obrekoval izdajo Sun-ove nove različice MySQL 5.1. Naslednje leto je podjetje Sun zapustil tudi Mårten Mickos in tako je Sun ostal brez talentov, vizionarjev, ustanoviteljev produkta MySQL. V tem letu se je Sun dogovoril z podjetjem Oracle, da jih kupi za 7.4 milijarde dolarjev in tako je podjetje Oracle končno prišlo do tako željene podatkovne zbirke MySQL. Status produkta MySQL pa se je pod okriljem novega lastnika zelo hitro spremenil - poslabšal. Zaradi takega statusa in slabega razvoja podatkovne zbirke MySQL pod okriljem podjetja Oracle, se je tako ustanovitelj MySQL-a, "Monty" odločil, da razvije novo podatkovno zbirko MariaDB, ki jo je prav tako kot MySQL poimenoval po svoji hčerki, najmlajši hčerki, ki ji je ime Maria.

Cilj diplomske naloge je predstavitev poteka migracije podatkov iz podatkovne zbirke MySQL na podatkovno zbirko MariaDB in Percona Server. Prikazan je praktičen primer aplikacije, ki smo ga naredili z razvojnim orodjem Microsoft Visual Studio 2010. Aplikacija naredi poizvedbo iz testne baze »classicmodels«, ki jo najprej namestimo na podatkovno zbirko MySQL, potem pa na podatkovno zbirko MariaDB.

Diplomska naloga obsega šest poglavij. V prvem poglavju se dotaknemo zgodovine shranjevanja podatkov in predstavimo kronološki potek razvoja podatkovne zbirke MySQL. V drugem poglavju – *Predstavitev podatkovnih zbirk* - predstavimo podatkovne zbirke na katere se osredotočamo skozi celo diplomsko nalogo, to so MySQL, MariaDB in Percona Server. V tretjem poglavju – *Primerjava podatkovnih baz* – opisujemo lastnosti podatkovnih zbirk in jih med seboj primerjamo, osredotočimo se predvsem na stroj za upravljanje s podatki InnoDB in XtraDB. V četrtem poglavju – *Migracija podatkov* - teoretično opisujemo pravilen potek migracije podatkov na podatkovno zbirko MariaDB in Percona Server. V petem poglavju – *Praktična izvedba migracije na podatkovno zbirko MariaDB* – prikažemo na testnem primeru, s pomočjo testne baze in aplikacije potek migracije na podatkovno zbirko MariaDB. V zadnjem, zaključnem poglavju so povzete naše sklepne ugotovitve.

# 2 Predstavitev podatkovnih zbirk

Podatkovne baze so pomemben del informacijske tehnologije, kajti v njih se hranijo najrazličnejši podatki, podatki o nakupih, izdanih računih, artiklih ki jih ima podjetje v skladišču, kupcih, vremenskih razmerah in sploh vsem, kar ima možnost vpliva na bodoče poslovanje. Podatkovne baze so se pojavile zaradi potrebe po hitrem dostopu do teh podatkov, saj nam shranjeni podatki iz preteklosti omogočajo, da lahko premišljeno odločamo o prihodnosti. Zgradbo podatkovnih baz delimo na dva dela in sicer na meta podatkovno bazo in fizično podatkovno bazo. Meta podatkovna baza shranjuje opise fizičnih podatkov, kje se ti podatki nahajajo v zunanjem pomnilniku, kaj pomenijo ti podatki in katerim uporabnikom so dostopni. Fizična podatkovna baza pa shranjuje fizične vrednosti podatkovnih elementov, ki se nanašajo na lastnosti opazovanih objektov. Poznamo več vrst podatkovnih baz, hierarhične in mrežne, ki danes niso več v uporabi, objektne, ki se uporabljajo samo v posebne namene in relacijske, ki so v današnjih časih še vedno najbolj pogosto uporabljene, kljub temu, da se pojavljajo namenske rešitve za upravljanje s podatki, ki sodijo v skupino NoSQL podatkovnih zbirk.

Podatkovne zbirke oz. baze, ki smo jih uporabljali in analizirali v diplomski nalogi so relacijske podatkovne baze, njihove podrobnejše lastnosti, značilnosti smo opisali v naslednjih podpoglavjih.

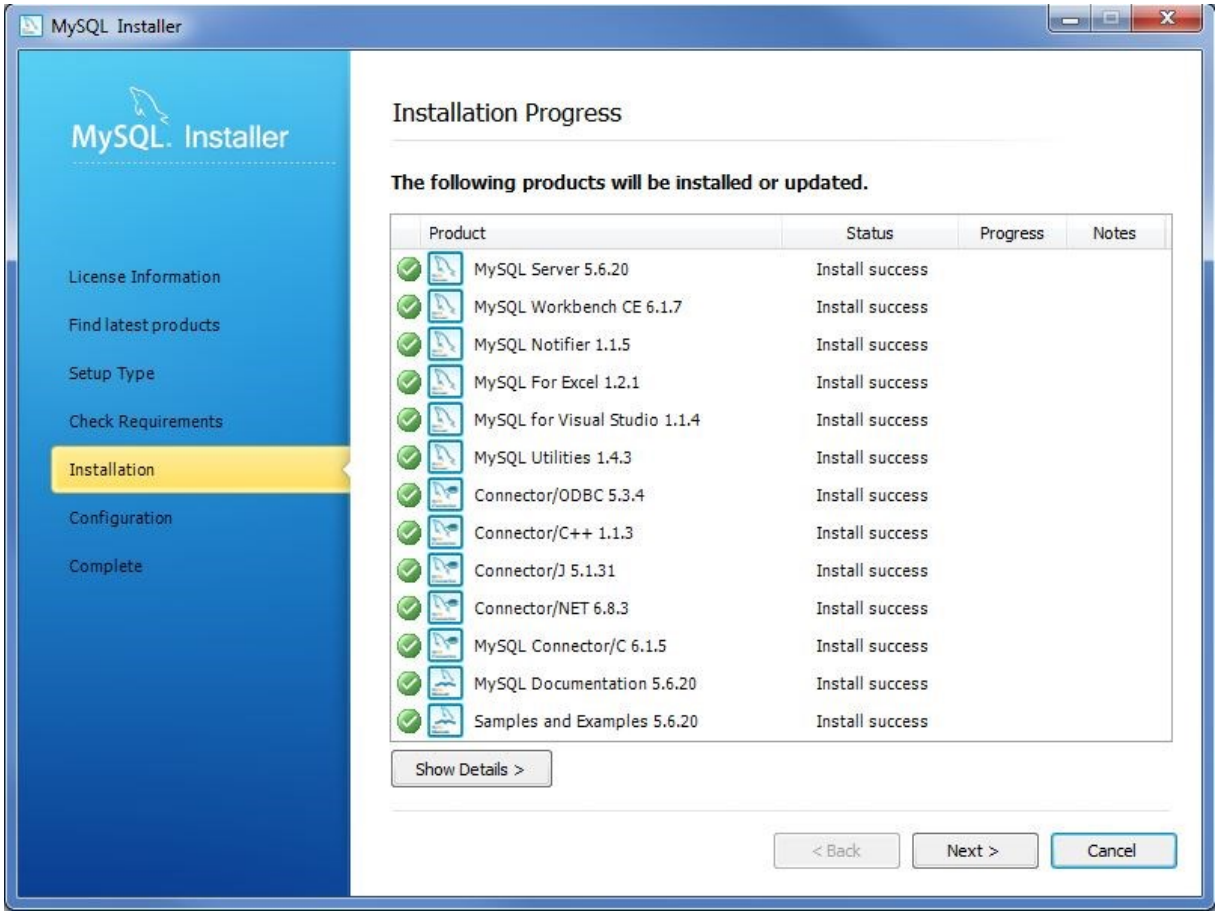
## 2.1 MySQL

Odprtokodna podatkovna baza MySQL je najbolj razširjena po Svetu, uporabljajo jo tako zasebni uporabniki, ki nimajo veliko povpraševanj, dinamičnih pogledov, posodobitev na dan, kot tudi velika, svetovno znana podjetja, kot so npr. Google, Wikipedia, Amazon.com, Yahoo!, NASA, Friendster, ki imajo npr. tudi več kot 1.3 milijona posodobitev na dan in 11 tisoč povpraševanj na sekundo (Wikipedia). Na podlagi teh številk lahko rečemo, da gre za zelo zmogljivo podatkovno zbirko, ki je tudi zelo prilagodljiva podatkovna zbirka saj deluje na mnogih platformah kot so GNU/Linux, Mac OS X, Novell NetWare, Solaris, SunOS, seveda tudi na večini različic Windows in še mnogo drugih platformah. Značilnost MySQL-a je tudi ta, da vsebuje različne podatkovne stroje (MyISAM, Merge, InnoDB, BDB, Memory/heap, Cluster, Federated, Archive, CSV, Blackhole), ki dovoljujejo izbiro tistega podatkovnega

stroja, ki je najučinkovitejši v dani situaciji. To je značilnost, s katero se nekatere druge podatkovne baze ravno ne morejo pohvaliti.

Omenimo lahko tudi še nekatere druge pomembne lastnosti, kot so: podatkovna zbirka MySQL je napisana v programskem jeziku C in C++, ima možnost nastavljanja »key_buffer_size« s katerim lahko izboljšamo upravljanje z indeksi pri bralno – pisalnih operacijah, testirana je z veliko vrstami različnih orodij.

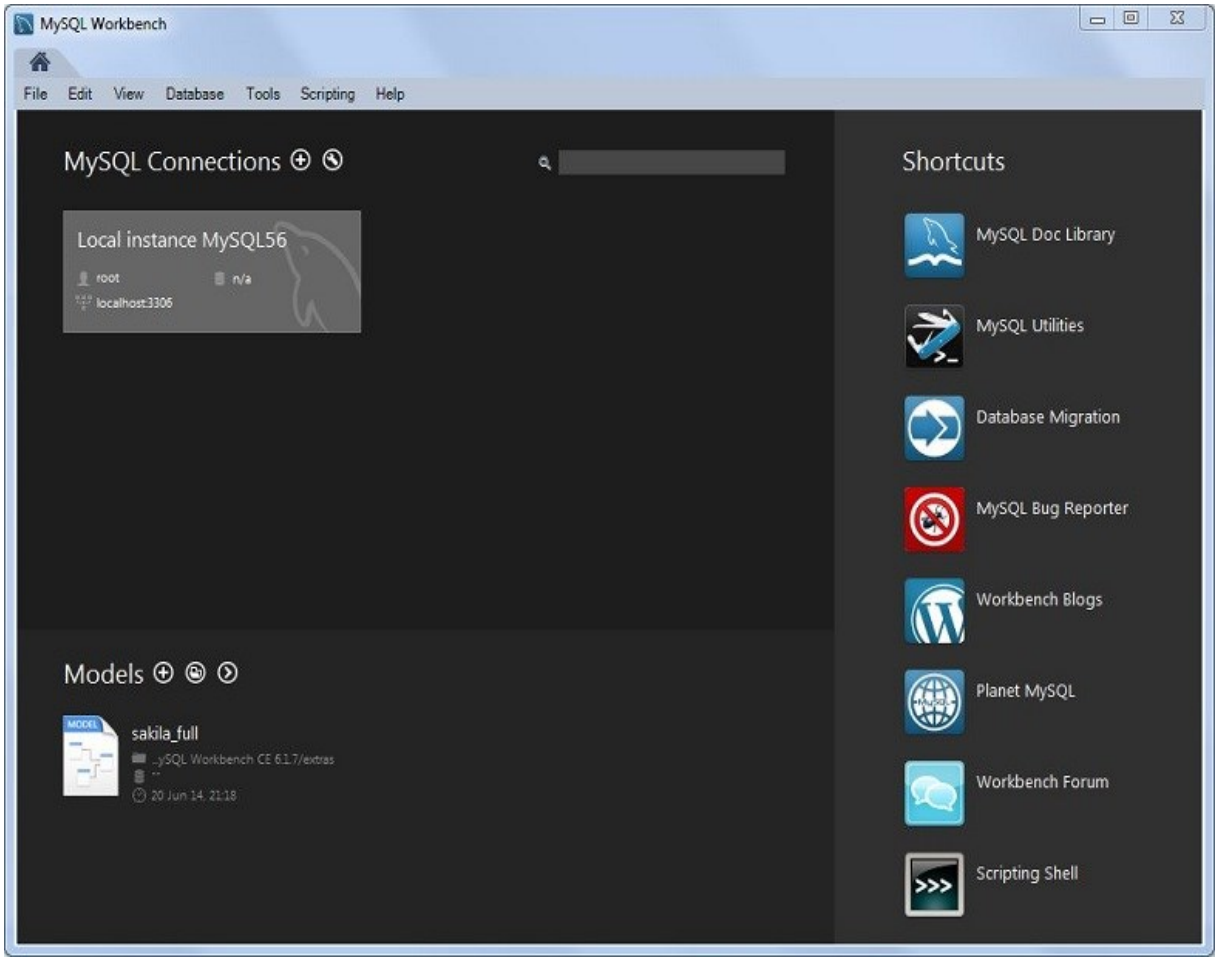
Za namestitev podatkovne zbirke MySQL smo si najprej prenesli MySQL Installer, ki je dosegljiv na spletni strani [3]. Namestitev podatkovne zbirke MySQL je s pomočjo MySQL Installer (Slika 2.1) za uporabnika zelo enostavna, kajti paziti ni potrebno tudi na to ali uporabljamo 32 ali 64 bitni operacijski sistem.




Slika 2.1: MySQL Installer in njegovi namestitveni produkti.

Pri namestitvi podatkovne zbirke MySQL se s pomočjo MySQL Installer namestijo poleg produkta MySQL Workbench, ki ga bomo omenili in opisali malo kasneje, tudi vsi potrebni povezovalniki (za ODBC, NET, C, C++, J), potrebne knjižnice za komunikacijo s programskim orodjem Visual Studio in Excel, dokumentacija MySQL itn. Pomemben produkt, za katerega

je narejen GUI s katerim lahko administriramo strežnik, izdelujemo modele EER, izvajamo poizvedbe SQL, delamo nove podatkovne baze, nove tabele, funkcije, se imenuje MySQL Workbench (Slika 2.2), ki smo ga za izvedbo testa migracije podatkov v petem poglavju uporabljali tudi mi.



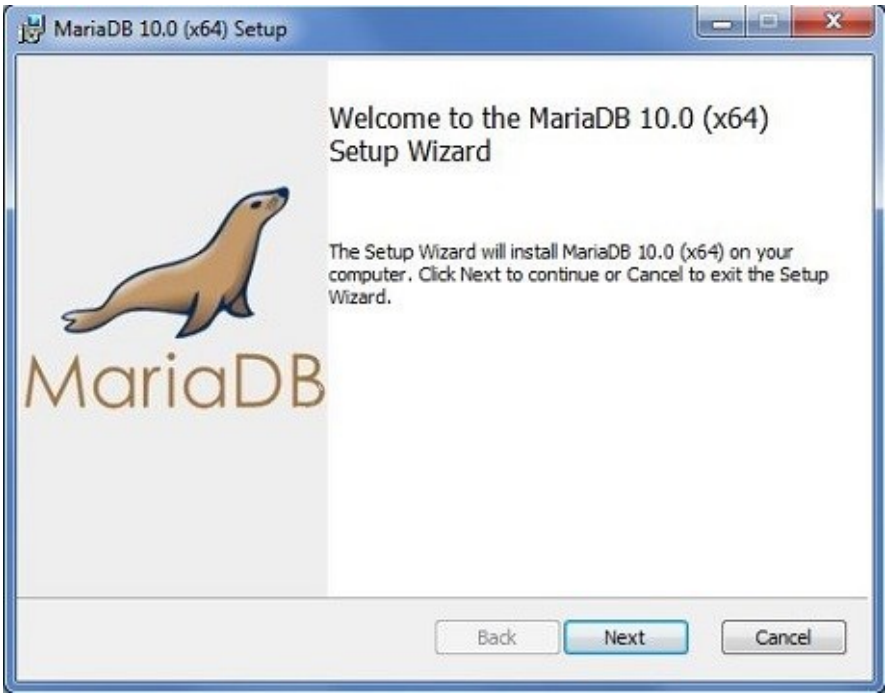

Slika 2.2: MySQL Workbench.

## 2.2 MariaDB

Odprtokodna podatkovna zbirka MariaDB je relativno nova, njena prva različica je bila izdana leta 2009, vendar zaradi tega ni nič manj popularna saj gre v bistvu za naslednjo generacijo njene »zvezdniške« predhodnice, podatkovne zbirke MySQL. Od svoje prve različice je podatkovna zbirka MariDB v kratkem času pridobila ogromno zvestih uporabnikov, tako rekoč več kot katerakoli druga podatkovna baza. Danes jo uporablja na deset tisoče spletnih strani, velikih in majhnih. Uporabljajo jo različna podjetja po svetu, iz različnih področij industrije, z več sto tisoč uporabniki. Podatkovna zbirka MariaDB je bila razvita kot »drop-in« nadomestek podatkovne zbirke MySQL, kar pomeni, da so funkcionalnosti, večina značilnosti, lastnosti,

enake kot pri podatkovni zbirki MySQL. Vsi ukazi, uporabljene knjižnice, vmesniki, aplikacijski programski vmesniki, ki so v podatkovni zbirki MySQL, obstajajo tudi v podatkovni zbirki MariaDB. MariaDB je tako kot MySQL razvita za mnogo operacijskih sistemov. V primerjavi s podatkovno zbirko MySQL MariaDB uporablja podatkovne stroje Aria, TokuDB, PBXT, FederatedX, Cassandra, Spider in tudi XtraDB, ki vsebuje vse kar vsebuje podatkovni stroj (ang. storage engine) InnoDB, poleg tega pa vsebuje še določene dodatke za skalabilnost (razširjenost), meritve in prilagodljivost. Ima tudi precej boljše performančne rezultate na več jedrih in ima tudi boljšo izrabo pomnilnika. Varnost, ki je v današnjih časih zelo pomemben dejavnik, je zelo pomembna tudi pri razvijalcih podatkovne zbirke MariDB, saj vzdržujejo svoj lastni niz varnostnih popravkov. Kadar odkrijejo kritična varnostna vprašanja, razvijalci kar najhitreje to popravijo in hitro izdajo novo različico, popravek podatkovne zbirke MariaDB. Značilnost podatkovne zbirke MariaDB je tudi ta, da je za nazaj kompatibilna z podatkovno zbirko MySQL.

Glede na značilnosti, lastnosti, ki smo jih raziskali, lahko sedaj tudi lažje razumemo odločitev nekaterih velikanov v današnjem informacijskem svetu Google, Wikipedia, Red Hat, ki so že napovedali prehod oz. migracijo podatkov iz podatkovne zbirke MySQL na podatkovno zbirko MariaDB. Za namestitev podatkovne zbirke MariaDB na računalnik (operacijski sistem Windows), je potrebno prenesti MSI Installer, ki je dosegljiv na spletu [2]. Glede na to koliko bitni (32 ali 64) operacijski sistem uporabljamo, moramo izbrati pravo različico MSI paketa. Pri naši izvedbi migracije v petem poglavju smo izbrali 64 bitno različico (Slika 2.3).

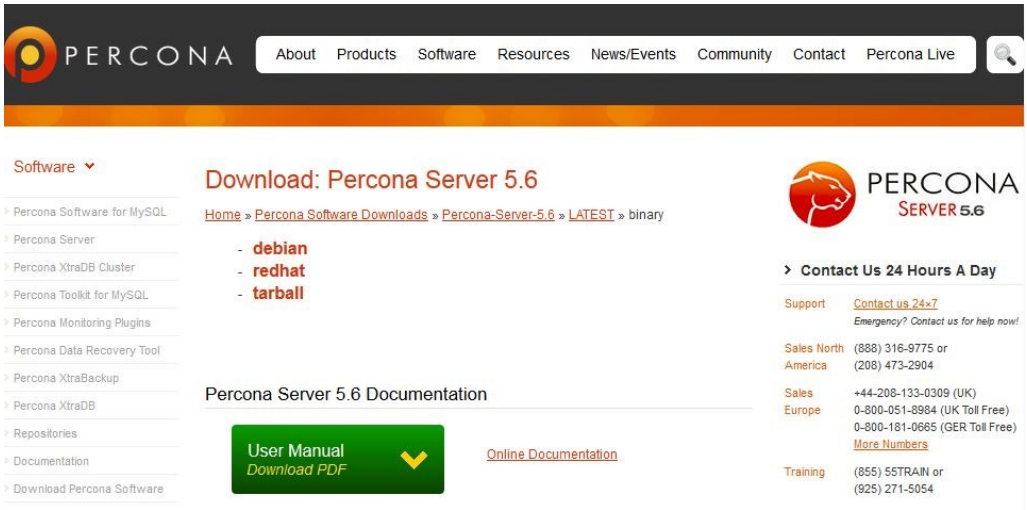



Slika 2.3: 64 bitni MSI.

## 2.3 Percona Server

Tako kot podatkovna zbirka MariaDB je tudi Percona Server razvita kot izboljšani »drop-in« nadomestek podatkovne zbirke MySQL. Ponuja revolucionarne značilnosti, zmogljivosti, skalabilnost in instrumentalnost. Percona Server je zasnovan za optimizacijo zmogljivosti. Skrivnost njegove zmogljivosti in stabilnosti so predvsem dolga leta testiranj, meritev ter razvoj podatkovnih strojev InnoDB in XtraDB. Izraz instrumentalnost, ki smo ga prej omenili, smo ga uporabili zato, ker je Percona Server najbolj merljiv strežnik. Z njim lahko vidimo podrobno beleženje poizvedb (statistične podatke o zaklepanjih na poizvedbo), učinkovitost in števec dostopov (glede na tabelo, indeks, uporabnika, gostitelja) in tudi statistiko odzivnih časov za merjenje uspešnosti (ne le števila operacij). Eden izmed revolucionarnih zmogljivosti je tudi skalabilnost. Kot primer lahko povemo, da pred leti podatkovna zbirka MySQL ni mogla učinkovito uporabljati štiri jederne procesorje in je bila omejena na 100 diskovnih operacij na sekundo, danes pa Percona Server lahko učinkovito uporablja tudi več kot 48 jedrne procesorje, več kot večina vseh strežnikov in lahko z ustreznim diskovjem doseže tudi več sto tisoč operacij na sekundo. Percona Server nam v primerjavi z MySQL in MariaDB zagotavlja najboljše rezultate zmogljivosti v različnih scenarijih, bistveno boljšo učinkovitost s 150 sočasnimi niti in celo do štiri-krat boljšo zmogljivost s 1000 ali več nitmi. Funkcija »ThreadPool«, ki jo uporablja Percona Server bistveno pripomore k zmogljivosti v bralno - pisalnih (ang. read - write) situacijah. Naj omenimo še, da je Percona Server tako kot podatkovna zbirka MariaDB, nazaj kompatibilen s podatkovno zbirko MySQL.

Pomembno je omeniti, da Percona Server ni razvit za operacijske sisteme Windows, namestiti ga je mogoče samo na operacijske sisteme, ki uporabljajo Linuxovo jedro (Slika 2.4).




Slika 2.4: Spletna stran[4] za prenos namestitve Percona Server.

Večja podjetja in spletne strani, ki uporabljajo Percona Server so Flickr, LinuxQuestions.org, Taba.ru, Wikia, Zimbio, ShermansTravel, Opera, Lionseek, Bluehost, ChinaNetCloud, MediaLayer in še mnogi drugi.

# 3 Primerjava podatkovnih baz

V prejšnjem poglavju omenjene podatkovne zbirke, MySQL, MariaDB in Percona Server smo primerjali med seboj in odkrili veliko enakosti glede kompatibilnosti (združljivosti) in tudi raznolikosti.

## 3.1 Primerjava lastnosti podatkovnih baz

Pri primerjanju lastnosti podatkovnih zbirk, smo se osredotočili predvsem na njihovo uporabo podatkovnih strojev, podvajanje (ang. replication) in splošne lastnosti. Za primerjavo splošnih lastnosti smo uporabili naslednja sodila: ali bodo podatkovni zbirki Percona Server in MariaDB kompatibilne z MySQL tudi v prihodnje, ali so binarne datoteke podatkovnih zbirk medseboj kompatibilne, ali so aplikacijski vmesniki, protokoli in konektorji (ang. connectors) med seboj kompatibilni, ali so podatkovne zbirke kompatibilne z podatkovnim strojem InnoDB oz. ali lahko uporabljajo podatkovni stroj InnoDB, ali ima določena podatkovna zbirka možnost uporabe podatkovnega stroja XtraDB. Naj omenimo še dve sodili: ali ima podatkovna zbirka izboljšane algoritme za »okrevanje po nesreči« (ang. crash recovery) in ali podatkovna zbirka uporablja podvojevanje z večnitnimi podrejenimi sistemi (ang. multi-threaded slave) (Slika 3.1).

| | Oracle MySQL 5.5 (InnoDB) | Percona Server 5.6 (InnoDB) | MariaDB 10 (XtraDB) |
|---|---|---|---|
| **splošne informacije** | **MySQL** | **Percona** | **MariaDB** |
| v prihodnje MySQL kompatibilni | da | da | da |
| binarne datoteke so medsebojno kompatibilne | da | da | da |
| API-ji, protokoli in konektorji so med seboj kompatibilni | da | da | da |
| **InnoDB** | **MySQL** | **Percona** | **MariaDB** |
| kompatibilnost z InnoDB | da | da | da |
| razpoložljivost XtraDB | ne | da | da |
| posebne izboljšave InnoDB | da | ne | da |
| bolj učinkovita uporaba več jedernih procesorjev | da - (z alokacijo pomnilnika) | da | da - (z uporabo jemalloc) |
| popravki vhodno - izhodne (I/O) skalabilnosti | da | da | da |
| izboljšani algoritmi za Crash Recovery | da | da | da |
| **podvajanje (replication)** | **MySQL** | **Percona** | **MariaDB** |
| podvajanje z večnitnimi slave-i | da - (za podatkovno bazo) | da - (za podatkovno bazo) | da - (za tabelo) |
| optimizacija vrstičnega podvajanja | da | ne | ne |
| Multi-source podvajanje | ne | ne | da |

Slika 3.1: Primerjava pomembnih lastnosti podatkovnih zbirk.

Raziskali smo tudi nekatere druge lastnosti med podatkovni zbirkami: izboljšave optimizatorja (ang. optimizer enhancements), NoSQL vmesnik, optimizacija urejanja datotek (ang. file sort optimization), optimizacija stikov (JOIN), izboljšave plana poizvedbe (EXPLAIN plan), optimizacija dostopov do diska in optimizacija vgnezdenih poizvedb (ang. subquery optimizations) (Slika 3.2).

| | Oracle MySQL 5.5 (InnoDB) | Percona Server 5.6 (InnoDB) | MariaDB 10 (XtraDB) |
|---|---|---|---|
| **drugo** | **MySQL** | **Percona** | **MariaDB** |
| izboljšave optimizatorja | da | da | da - (verjetno najboljši) |
| NoSQL vmesnik preko predpomnilnika (memcached) | da - (memcached) | ne | da - (cassandra in memcached) |
| optimizirano urejanje datotek | da | da | da |
| optimizacija stikov | ne | ne | da |
| izboljšave plana poizvedbe | da | da | da |
| optimizacija dostopov do diska | da | da | da - (verjetno najboljša) |
| optimizacija vgnezdenih poizvedb | da | da | da |

Slika 3.2: Primerjava ostalih lastnosti podatkovnih zbirk.

Raziskali pa smo tudi primerjavo nekaterih novejših operacij med podatkovnima zbirkama MySQL in MariaDB. Na tem področju smo odkrili kar nekaj razlik (Slika 3.3).

| | Oracle MySQL 5.5 (InnoDB) | MariaDB 10 (XtraDB) |
|---|---|---|
| **operacije** | **MySQL** | **MariaDB** |
| izboljšano raziskovanje tabel | ne | da |
| SHOW PLUGINS SONAME | ne | da |
| ukaz SHUTDOWN | ne | da |
| ubije query za določen query ID | ne | da |
| ukaz SHOW EXPLAIN | ne | da |
| izboljšana sporočila o napakah | ne | da |
| online ALTER TABLE | da | da |

Slika 3.3: Primerjava nekaterih operacij med MySQL in MariaDB.

Operacija »izboljšano raziskovanje tabel«, ki jo podpira podatkovni stroj FederatedX, nam omogoča, da uporabljamo tabele za katere ni bil eksplicitno pognan noben »CREATE TABLE« stavek. Operacija »SHOW PLUGINS SONAME« nam poda informacije o statusu namestitve nameščenih vtičnikov (ang. plug-in). Ukaz »SHUTDOWN« v SQL sintaksi elegantno ugasne strežnik in ima enake pravice kot »mysqladmin shutdown«. Operacija »Kill query by query ID« omogoča administratorju, da ubije eno določeno poizvedbo (ang. query) niti, ki se izvaja, ne da bi ubil povezavo z nitjo (ang. thread). Ukaz »SHOW EXPLAIN« je v veliko pomoč razvijalcem saj nam prikaže vse poizvedbe ki se izvajajo. Operacija »izboljšana sporočila o napakah« je

prav tako zelo uporabna za razvijalce saj vse napake vsebujejo opise. Operacija »online ALTER TABLE« pa nam omogoča, da se pri spreminjanju sheme tabele le-ta ne zaklene.

Ker uporablja podatkovna zbirka MySQL podatkovni stroj InnoDB, podatkovna zbirka MariaDB pa podatkovni stroj XtraDB, smo se odločili, da še podrobneje raziščemo lastnosti obeh, kar je predstavljeno v naslednjih podpoglavjih.

## 3.2 Podatkovni stroj InnoDB

Podatkovni stroj InnoDB je splošno namenski podatkovni stroj, ki ponuja kompromis med visoko zanesljivostjo in zmogljivostjo. V različici podatkovne zbirke MySQL 5.5 naprej je tudi privzeti podatkovni stroj. Njegove glavne prednosti so, da je njegov jezik za manipulacijo s podatki (DML) skladen z modelom ACID, tako da transakcije za zagotavljanje varnosti podpirajo opcije kot so izročitev, povrnitev in okrevanje po zrušitvi. Za zagotavljanje sočasnega dostopa do podatkov in boljše zmogljivosti, uporablja InnoDB zaklepanje vrstic (ang. row level locking). Tabele podatkovnega stroja InnoDB razporejajo podatke na disk tako, da optimizirajo poizvedbe izvedene po primarnih ključih. Za zagotavljanje podatkovne celovitosti podpira omejitve tujih ključev. Vsaka operacija vstavljanja, posodobitve ali izbrisa je preverjena, s čimer se zagotovi, da med njimi ne prihaja do nekonsistentnosti v različnih tabelah. Podatkovni stroj InnoDB je bil načrtovan za maksimalno zmogljivost pri procesiranju velikih količin podatkov. Izkoristek procesorja je višji kot pri katerem koli drugem podatkovnem stroju relacijske podatkovne baze.

Raziskali smo nekatere pomembne lastnosti podatkovnega stroja InnoDB. Spodaj napisane informacije smo dobili na spletni strani[11]:

- *dvojnopisni medpomnilnik (ang. doublewrite buffer)* - Pred zapisom strani na datoteko, podatkovni stroj InnoDB podatke najprej zapiše na sosednji prostor tabel (ang. a contiguous tablespace area), poimenovan dvojnopisni medpomnilnik. Šele po tem zapisu začne podatkovni stroj InnoDB s pisanjem strani na končno lokacijo v datoteki. Če medtem pride do sesutja operacijskega sistema, pri čemur pride do stanja pretrgane strani*, podatkovni stroj InnoDB lahko kasneje poišče obstoječo veljavno kopijo strani v dvojnopisnem medpomnilniku.

  * *pretrgana stran* je stanje napake, ki nastane kot kombinacija konfiguracije vhodno - izhodnih naprav in napake strojne opreme. Če se podatki zapisujejo v kosih, manjših od posamezne strani podatkovnega stroja InnoDB (privzeto 16KByte), lahko strojna napaka (ang. hardware failure) med pisanjem pripelje do tega, da je le del strani shranjene na disku. Dvojnopisni medpomnilnik preprečuje to možnost.

- *indeksi gruče in sekundarni indeksi* - Vsaka tabela podatkovnega stroja InnoDB vsebuje poseben indeks, imenovan indeks gruče (ang. clustered index), v katerega lahko shranimo podatke vrstic. Običajno je indeks gruče enakovreden primarnemu ključu. Deluje po naslednjem principu: ko definiramo primarni ključ v tabeli, ga InnoDB uporabi tudi za gručen indeks. Primarni indeks kreiramo za vsako tabelo, ki jo ustvarimo. Če logično unikaten stolpec ali skupina stolpcev ne obstajata, se doda nov stolpec s samodejnim prirastkom (ang. autoincrement), katerega vrednosti se polnijo avtomatsko. V kolikor ne definiramo primarnega ključa, se za gručen indeks določi prvi enoličen indeks, katerega vsi ključni stolpci so neničelni. V kolikor nimano niti primarnega ključa niti ne izpolnjujemo pogojev za gručen indeks, podatkovni stroj InnoDB sam generira skriti gručen indeks, ki vsebuje zaporedno številko vrstice. Vrstice so urejene po oznaki, ki jih podatkovni stroj InnoDB dodeli tabeli, kateri pripadajo. Tako oznako sestavlja polje dolžine 6 bajtov, ki se z večanjem tabele samodejno povečuje. S tem so vrstice dejansko urejene po vrstnem redu vstavljanja. Dostop do vrstic preko indeksa gruče predstavlja pohitritev zato, ker iskanje privede direktno do strani, ki vsebuje vse podatkovne vrstice (ang. data row). V kolikor je ta tabela velika, uporaba metode indeksa gruče pogosto vodi do zmanjšanega obsega vhodno-izhodnih operacij v primerjavi z drugimi organizacijskimi metodami, ki shranjujejo podatkovne vrstice na eno stran, indeksne zapise pa na drugo stran. V tej alineji lahko še povemo nekaj o fizični strukturi indeksa (ang. index) podatkovnega stroja InnoDB. Vsi indeksi so B-drevesa (ang. binary tree), pri čemer so zapisi o indeksih shranjeni v listih. Privzeta velikost indeksne strani je 16KByte. Ko vstavljamo nov zapis, želi podatkovni stroj InnoDB vedno zagotoviti proste 1/16 strani za prihajajoča vstavljanja in obnovitve zapisov. Če jih shranjujemo zaporedno (padajoče ali naraščajoče), je novo nastala indeksna stran približno 15/16 polna. Če vstavljamo po naključnem vrstnem redu, bodo strani nekje med 1/2 in 15/16 polne. V kolikor ta faktor pade pod 1/2, poskuša podatkovni stroj InnoDB skrčiti indeksno drevo z namenom sprostitve strani. Velikost strani lahko v splošnem nastavimo ob kreaciji, vendar je kasneje ne moremo več spreminjati. Možne velikosti so sicer 16KByte, 8KByte, 4KByte.

- *fizična struktura vrstic* - Fizična struktura tabele je odvisna od formata vrstice specificiranega ob tvorbi tabele. Za datoteke je privzet format »Antelope« in njegov vrstični format »Compact«. Omogočena sta tudi starejši »Redundant« in novejši »Barracuda«, slednji omogoča dinamične in kompresirane (ang. compressed) vrstične formate. Format »Compact« zmanjša velikost, potrebno za shranjevanje vrstic za približno 20%, za svoje delovanje pa potrebuje nekaj več procesorske moči. V kolikor je vzrok počasnosti delovanja nezadostna velikost pomnilnika, potem bo »Compact« verjetno pospešil izvedbo. V določenih redkih primerih, ko smo omejeni s hitrostjo procesorja pa

je »Compact« počasnejši. Vrstice v InnoDB tabelah, ki uporabljajo format »redundant« imajo sledeče karakteristike: vsak zapis indeksa vsebuje glavo dolžine 6Byte, katera povezuje zaporedne zapise, uporablja pa se tudi pri zaklepanju vrstic (ang. row-level locking). Zapisi v indeksu gruče vsebujejo polja za vse uporabniško ustvarjene stolpce, poleg tega pa tudi oznako transakcije dolžine 6Byte in polje kazalca dolžine 7Byte. V tabelah brez definiranega primarnega ključa, vsebuje vsak indeks gruče polje za oznako vrstice dolžine 6Byte. Vsak sekundarni indeks vsebuje vrednosti primarnih ključev primarnega indeksa (ang. clustered index). Zapis vrstice v InnoDB tabeli vsebuje kazalec za vsako polje zapisa. Če je skupna dolžina vseh polj v zapisu manjša od 128Byte, je kazalec velikosti 1Byte, sicer pa 2Byte. Tabelo teh kazalcev imenujemo imenik zapisov. Območje, kamor ti kazalci kažejo, imenujemo podatkovni del zapisa. Jezik SQL za »NULL« vrednost rezervira en ali dva bajta v imeniku zapisov. Poleg tega jezik SQL za »NULL« rezervira 0 bajtov v podatkovnem delu zapisa, če je datoteka shranjena v stolpcu nefiksne dolžine. Nasprotno v stolpcu fiksne dolžine rezervira fiksno dolžino stolpca v podatkovnem delu zapisa. S tem ko rezervira fiksno velikost »NULL« vrednosti, omogoča spremembo vrednosti stolpca iz »NULL« na »non-NULL«, brez da bi bila potrebna fragmentacija indeksne strani. Vrstice v InnoDB tabelah, katere uporabljajo vrstični format »Compact« imajo naslednje karakteristike: vsak indeksni zapis vsebuje glavo velikost 5Byte, le ta ima lahko pred seboj dodatno glavo spremenljive velikosti. Ta glava se uporablja za povezovanje zaporednih zapisov ter za vrstične zaklepe. Spremenljivi del glave vsebuje bitni vektor za označevanje ničelnih stolpcev. Če je število stolpcev v indeksu, katerih vrednosti so lahko »NULL«, enako N, potem je za bitni vektor potrebnih vsaj N/8 bajtov. Stolpci, kateri imajo vrednost »NULL« zasedajo le prostor v tem bitnem vektorju. Del spremenljive dolžine prav tako vsebuje dolžine stolpcev z spremenljivo dolžino. Vsaka dolžina potrebuje 1 ali 2 bajta, odvisno od največje dolžine stolpca. V kolikor so vsi stolpci v indeksu neničelni in dolžine fiksne, potem glava zapisa nima dodatnega dela spremenljive dolžine. Glavi zapisa sledijo podatki neničelnih stolpcev. Zapisi v indeksu gruče vsebujejo polje za vse uporabniško definirane stolpce. Poleg tega vsebujejo tudi polje za oznako transakcije dolžine 6Byte in polje kazalca dolžine 7Byte. V kolikor tabela nima določenega primarnega ključa, potem vsebuje vsak indeks gruče tudi polje za oznako vrstice dolžine 6Byte. Vsak zapis sekundarnega indeksa vsebuje vsa polja primarnega ključa, ki so definirana za ključ indeksa gruče, vendar niso v sekundarnem indeksu. Če je katero od polj primarnega ključa spremenljive dolžine, ima glava zapisa vsakega sekundarnega indeksa poseben del spremenljive dolžine za beleženje njihove dolžine, tudi če je sekundarni indeks definiran na stolpcih fiksne dolžine. Lokalno, InnoDB shranjuje stolpce oznak (ang. characters) fiksne dolžine in fiksne širine, npr. »CHAR(10)« v formatu fiskne dolžine. InnoDB ne odbija praznih pripon v stolpcih tipa »VARCHAR«.

Po drugi strani pa poskuša InnoDB stolpce »UTF-8 CHAR(N)« shraniti v N bajtov s krajšanjem pripone. V formatu »Redundant« bi taki stolpci zasegli 3xN bajtov.

- *InnoDB vhodno-izhodne operacije na disku in upravljanje datotečnega prostora (ang. file space management)* – podatkovni stroj InnoDB uporablja asinhrone vhodno - izhodne dostope do diska, kjer je to možno. Poteka tako, da ustvari več niti za upravljanje IO operacij, s čimer dovoljuje drugim operacijam baz, da se nadaljujejo, medtem ko se IO operacije še vedno izvajajo. Na operacijskih sistemih Linux in Windows, InnoDB uporablja operacijski sistem in njegove knjižnice za izvajanje domorodnih asinhronih IO operacij. Na drugih platformah InnoDB še vedno uporablja IO niti, vendar lahko niti čakajo na končanje IO zahtevkov, ta tehnika se imenuje simuliran asinhroni IO. V kolikor lahko InnoDB z pravšnjo verjetnostjo določi, da bo določene podatke potreboval kmalu, se izvrši operacija predbranja (ang. read ahead), katera premakne podatke v medpomnilniški bazen, tako da so potem na voljo direktno iz pomnilnika. Izvedba več razmeroma velikih poizvedb je lahko precej bolj optimalna kot izvedba mnogo manjših, razpršenih poizvedb. Podatkovni stroj InnoDB vsebuje dve hevristiki predbranja: zaporedno predbranje in naključno branje. Če pri zaporednem predbranju InnoDB zazna, da je vzorec dostopov do segmentov v podatkovni tabeli zaporeden, že vnaprej naslovi skupke (ang. batch) branj strani podatkovnih baz na IO sistem. Če pri naključnem branju podatkovni stroj InnoDB zazna, da so nekatera območja v prostoru tabel brana cela v medpomnilniškem bazenu, naslovi preostanek branj na IO sistem. Podatkovne datoteke, ki jih definira uporabnik predstavljajo prostor tabel InnoDB-ja. Datoteke so logično sestavljene tako, da tvorijo prostor tabel. Tu se obrezovanje ne uporablja, hkrati pa se ne da definirati, kje so locirane naše tabele znotraj prostora tabel. V novo nastalem prostoru tabel, podatkovni stroj InnoDB zaseže - priskrbi prostor, ki se začne s prvo podatkovno datoteko. Da se izognemo težavam, ki nastanejo s tem, da se vse tabele in indeksi hranijo znotraj istega sistema tabel, lahko uporabimo možnost »innodb_file_per_table«, le ta shrani vsako novonastalo tabelo v svojem prostoru tabel (s končnico .ibd). Pri tem načinu pride do manj fragmentacij, delitev znotraj diskovne datoteke (ang. disk file), prav tako se, ko je določena tabela odsekana, prostor vrne operacijskemu sistemu, namesto da bi bil še vedno zasežen s strani prostora tabel podatkovnega stroja InnoDB.

- *strani, obsegi, segmenti in prostori tabel* - Vsak prostor tabel sestoji iz strani podatkovne baze in ima enako velikost strani. Privzeto imajo vsi prostori strani velikost strani 16KByte. To velikost lahko spremenimo, tako da nastavimo »innodb_page_size« na 8KByte ali 4KByte. Strani so združene v posamezne obsege velikosti 1MByte (64 zaporednih 16KByte strani ali 128 8KByte strani ali 256 4KByte strani). Datoteke znotraj prostora tabel se v podatkovnem stroju InnoDB imenujejo segmenti (ločujemo jih od povrnjenih

(ang. rollback) segmentov, ki vsebujejo več segmentov prostorov tabel). Ko se povečuje velikost segmenta znotraj prostora tabel, mu podatkovni stroj InnoDB dodeli prvih 32 strani eno za drugo. Potem InnoDB dodeli celoten obseg segmentu. Naenkrat lahko dodeljuje največ 4 obsege večjemu segmentu, s čimer se zagotovi dobra zaporednost podatkov. Vsakemu indeksu pripadata dva segmenta. Eden predstavlja nelistnata vozlišča v B-drevesu, drugo pa listnata vozlišča. Zagotavljanje sosednjih listov na disku omogoča boljše zaporedne IO operacije, ker so listi tisti, ki dejansko vsebujejo podatke tabel. Nekatere strani v prostoru tabel vsebujejo bitne preslikave (ang. bitmap) drugih strani, zato nekaj obsegov znotraj prostora tabel ne moremo alocirati segmentom kot celoto, temveč le kot posamezne strani. Ko želimo izvedeti količino prostega prostora v prostoru tabel, podatkovni stroj InnoDB vrne količino dejansko prostega prostora. Sicer pa vedno rezervira nekaj obsegov za čiščenje in ostale notranje procese, le ti obsegi niso vključeni v nezaseden prostor. Ob brisanju podatkov iz tabele, podatkovni stroj InnoDB pridobi ustrezne indekse v B-drevesu. Ali bo sproščeni prostor na voljo ostalim uporabnikom, je odvisno od tega ali vzorec izbrisov sprosti posamezne strani ali posamezne obsege znotraj prostora tabel. Brisanje tabele oz. brisanje vseh njenih vrstic, ki jo sestavljajo, bo zagotovo dalo na voljo prostor ostalim uporabnikom, pri čemer je vredno poudariti, da bodo izbrisane vrstice fizično odstranjene šele pri operaciji odpisa, katera pa se zgodi, ko le te nekaj časa niso bile uporabljene za povrnitev oz. skladno branje (ang. consistent read).

## 3.3 Podatkovni stroj XtraDB

Pri raziskovanju lastnosti, funkcij podatkovnega stroja XtraDB smo se osredotočili predvsem na njegove izboljšave glede na podatkovni stroj InnoDB[5]:

- *izboljšana skalabilnost pomnilniškega bazena* - Ta zmožnost razdeli en sam globalen InnoDB sistem predpomnilniškega bazena na več podsistemov z namenom preprečevanja ozkih grl v, kar ima lahko velik pomen, ko ni dovolj prostora za delovne strani v pomnilniku.

- *izboljšana skalabilnost vhodno-izhodnih sistemov* - Ker je InnoDb kompleksen podatkovni stroj za shranjevanje, mora biti primerno konfiguriran za dosego najboljših rezultatov. Nekatere opcije v podatkovnem stroju InnoDB v primerjavi z podatkovnim strojem XtraDB niso konfigurabilne. Cilj te funkcije je zagotavljanje večjega nabora opcij za podatkovni stroj XtraDB, npr. zmožnost spremembe velikosti datoteke log (ang. log file).

- *prilagodljive razpršene iskalne particije* - Adaptivni razpršeni indeks podatkovnega stroja InnoDB ima težave z ozkimi grli na več jedrnih sistemih, posebej v primerih, ko se izvaja

kombinacija bralnih in pisalnih poizvedb, katere potrebujejo preiskovanje sekundarnih indeksov. Ta funkcija razdeli prilagodljive razpršene indekse na več particij, da se izogne takim težavam. Ustvarjeno je število particij, določeno s spremenljivko »innodb_adaptive_hash_index_partitions«, razpršeni indeksi pa so dodeljeni vsaki izmed njih na podlagi oznake »index_id«.

- *podpora atomarnemu zapisu (ang. atomic write) za vhodno - izhodne naprave Fusion* - Atomarni zapis je lahko uporabljen namesto InnoDB-jevega dvojnopisnega predpomnilnika za zagotovitev, da bodo podatkovne strani podatkovnega stroja InnoDB zapisane na disk v celoti ali pa ne bodo zapisane. Ko so ti zapisi omogočeni, bo naprava poskrbela za zaščito podatkov pred delnimi zapisi. V kolikor je vključen dvojnopisni predpomnilnik, se atomarni zapis samodejno izključi. Ta funkcija pohitri operacije pisanja, saj podatkov ne potrebujemo več pisati dvakrat, s tem pa je okrevanje hitrejše.

- *izboljšana podpora za NUMA (ang. Non-uniform memory access)* - V primerih, kjer je medpomnilniški bazen večji od vozlišča, lahko začne sistem zamenjevati že alociran pomnilnik, tudi če je v drugih vozliščih še nezaseden prostor. To bi se zgodilo, če bi bila izbrana privzeta funkcija alokacije pomnilnika NUMA. V tem primeru bi sistem favoriziral eno vozlišče bolj kot drugo, kar lahko povzroči, da bi vozlišče ostalo brez pomnilnika. Spreminjanje alokacijske strategije na prepredajočo, pomeni, da se pomnilnik alocira po sistemu »round-robin« nad prostim vozliščem. To storimo z uporabo opcije »mysqld_safe numa_interleave«. Druga izboljšava alokacijske strategije je predalociranje strani v medpomnilniškem bazenu ob zagonu s spremenljivko »innodb_buffer_pool_populate«. To pripelje do tega, da se NUMA alokacijske odločitve opravljajo, ko je predpomnilnik prazen.

# 4 Migracija podatkov

V tem poglavju smo se osredotočili na pravilen potek migracije podatkov na novo podatkovno zbirko MariaDB in Percona Server. V naslednjih dveh podpoglavjih smo natančno opisali ta postopek. Pri pravilnem poteku migracije podatkov iz podatkovne zbirke MySQL na Mariadb smo se osredotočili na primer migracije na operacijskem sistemu Windows. Pri poteku migracije podatkov iz podatkovne zbirke MySQL na Percona Server pa smo se osredotočili na primer migracije na operacijskem sistemu Ubuntu.

## 4.1 Potek migracije na MariaDB

Pri migraciji podatkov iz podatkovne zbirke MySQL na MariaDB je pomembno predvsem to, da najprej analiziramo kompatibilnost različice podatkovne zbirke MariaDB, na katero želimo migrirati podatke, z različico MySQL, ki jo že imamo nameščeno. Pomembne nekompatibilnosti različic bomo omenili kasneje. Migracijo podatkov na podatkovno zbirko MariaDB lahko obravnavamo kot posodobitev (ang. update) podatkovne zbirke MySQL, kar pomeni, da nam sploh ni potrebno narediti varnostne kopije (ang. backup) podatkov. Pomembno pri tem je, da podatkovne zbirke MySQL seveda ne odstranimo (ang. uninstall), če jo odstranimo je nujno potrebno narediti varnostno kopijo podatkov. Priporočljivo je, da pred zagonom namestitvenega programa MSI za podatkovno zbirko MariDB ugasnemo (ang. shutdown) vse MySQL storitve, ki se izvajajo na strežniku terminalu. Po namestitvi podatkovne zbirke MariaDB se pokaže GUI, čarovnik (ang. wizard),orodje za posodobitev, nadgradnjo najdene že nameščene podatkovne zbirke MySQL, ki izvede ukaz »mysql_upgrade_service«. Prej omenjeno orodje za posodobitev zahteva polne skrbniške pravice (ang. full administrative privileges). Ukaz »mysql_upgrade_service« opravi vse potrebne korake pri pretvorbi podatkovne zbirke. Ker se v današnjih časih pojavljajo tako 32 kot tudi 64 bitni operacijski sistemi naj omenimo še to, da je možno imeti nameščeni obe različici podatkovne zbirke MariaDB na istem strežniku terminalu, torej lahko namestimo 32 bitno različico podatkovne zbirke MariaDB s pomočjo 32 bitnega MSI in 64 bitno različico s pomočjo 64 bitnega MSI.

Zelo pomembna novica za razvijalce aplikacij je, da po migraciji podatkov na podatkovno zbirko MariaDB v njihovih aplikacijah ni potrebno popravljati tako rekoč ničesar.

Raziskali smo nekaj primerov nekompatibilnosti-raznolikosti različic MySQL in različic MariaDB pri katerih je potrebno biti pozoren pri migraciji podatkov:

- *razlike med različicama MySQL 5.1 in MariaDB 5.1* - V tem primeru smo raziskali dve razliki. Ukaz »CHECKSUM TABLE« vrne drugačen rezultat kot pri MySQL, ker MariaDB ne ignorira vrednosti »NULL« v kolonah (ang. column) tabel, MySQL pa te vrednosti ignorira. MariaDB privzeto (ang. default) zavzame malo več pomnilnika kot MySQL, ker ima pri rokovanju (ang. handling) z začasnimi (ang. temporary) tabelami privzeto omogočen (ang. enabled) podatkovni stroj Aria. Če želimo, da MariaDB zavzame zelo malo pomnilnika (na račun učinkovitosti), lahko nastavimo vrednost »aria_pagecache_buffer_size« na 1MByte, privzeto je vrednost nastavljena na 128MByte.

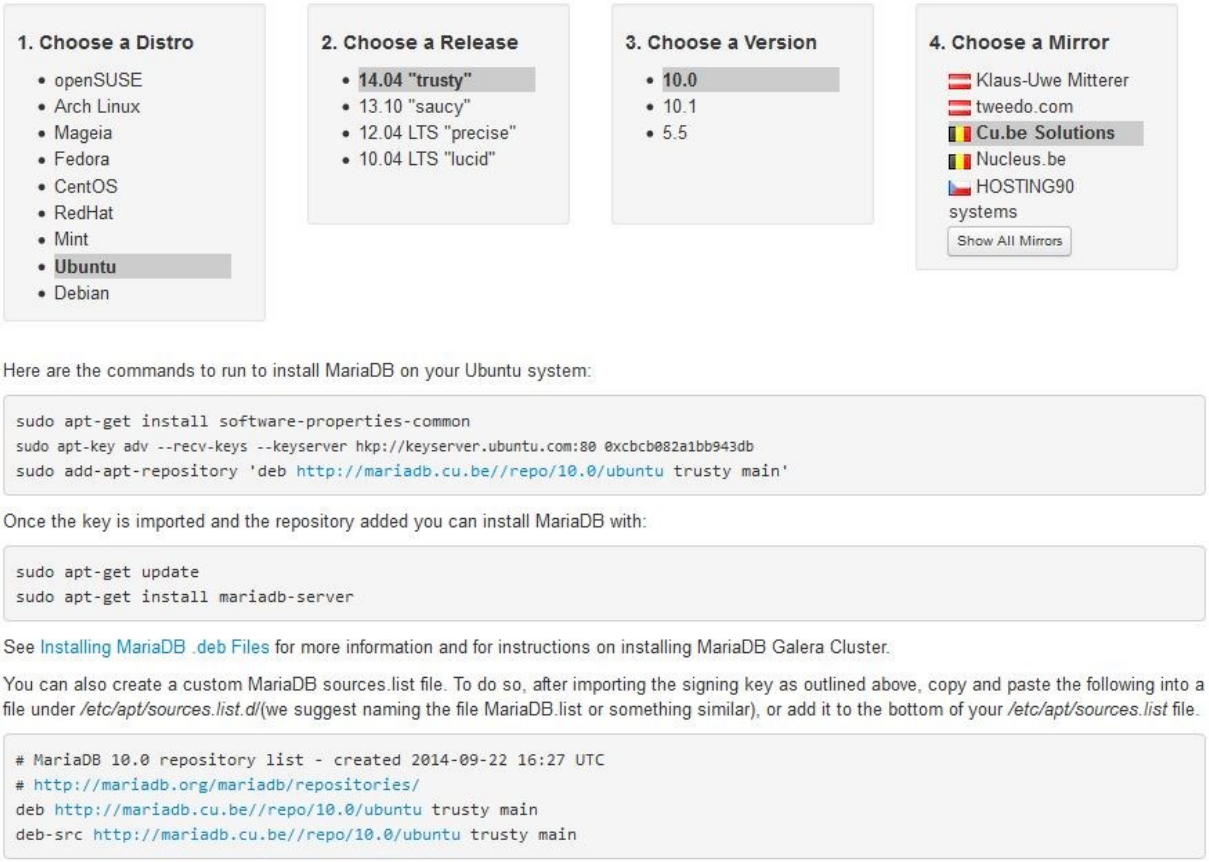
- *razlike med različicama MySQL 5.1 in MariaDB 5.2* - V tem primeru je še ena razlika več kot v prejšnji primerjavi različic MySQL 5.1 in MariaDB 5.2 in sicer v MariaDB je dodana nova vrednost »IGNORE_BAD_TABLE_OPTIONS«. Če ta vrednost ni definirana pri uporabi tabel ali polj in ni podprta s strani izbranega podatkovnega stroja, bo prišlo do napake. To lahko popravimo s pomočjo ukaza »mysql_upgrade«.

- *razlike med različicama MySQL 5.6 in MariaDB 10.0* - V tem primeru bomo omenili dve razliki. Prva je ta, da se ukaz »CREATE TABLE« izvede enako kot »CREATE OR REPLACE TABLE«. Druga razlika pa je ta, da ima MySQL 5.6 privzeto omogočeno shemo zmogljivosti (ang. performance schema), na račun zmogljivosti pa ima MariaDB shemo zmogljivosti privzeto onemogočeno (ang. disabled)

- *razlike glede MySQL proxy in Mariadb proxy* - MySQL odjemalčev (ang. client) API se je sposoben povezati (ang. connect) na strežnik MariaDB, MariaDB odjemalčev (ang. client) API pa nam bo sporočil, da se »MySQL-Proxy« ne izvaja. Da se izognemo tem težavam moramo na odjemalčevi (ang. client) ali strežniški (ang. server) strani onemogočiti (ang. disable) poročanja o napredku (ang. progress reporting).

Pri poteku migracije podatkov na podatkovno zbirko MariaDB na operacijskih sistemih Linux je izvedba migracije podobna kot pri migraciji na operacijskih sistemih Windows. Najprej prenesemo MariaDB namestitvene datoteke (ang. source tarball), namestimo (ang. install) podatkovno zbirko MariaDB, ustvarimo uporabnike (ang. users), imenike (ang. directory), inicializacijsko datoteko (ang. init file) in nastavimo dovoljenja (ang. permisions). Nato zaženemo podatkovno zbirko MariaDB in preverimo, če sta obe podatkovni zbirki zagnani in če sta, lahko začnemo premikati podatke iz podatkovne zbirke MySQL na MariaDB. Ko podatke prenesemo lahko potem podatkovno zbirko MySQL odstranimo (ang. uninstall) iz strežnika. Da bo namestitev podatkovne zbirke MariaDB in s tem tudi migracije podatkov še

bolj enostavna, je na spletni strani[8] točno prikazan seznam ukazov, ki jih moramo vnesti, da se bo podatkovna zbirka MariaDB pravilno namestila na določen operacijski sistem Linux. Izberemo najprej željeni operacijski sistem (Ubuntu, Debian, RedHat, Fedora, Mageia, Arch Linux, openSUSE, CentOS, Mint), različico operacijskega sistema in potem še željeno različico podatkovne zbirke. Primer ukazov namestitve podatkovne zbirke MariaDB 10.0 za operacijski sistem Ubuntu, različico 14.04 (Slika 4.1).




Slika 4.1: Namestitev podatkovne zbirke MariaDB 10.0 na Ubuntu 14.04.

## 4.2 Potek migracije na Percona Server

Podatkovno zbirko Percona Server je možno namestiti samo na operacijske sisteme Linux. Tako kot podatkovna zbirka MariaDB je tudi podatkovna zbirka Percona Server binarno združljiva, kompatibilna s podatkovno zbirko MySQL, zato lahko namestitev podatkovne zbirke Percona Server, tako kot podatkovno zbirko MariaDB, izvedemo kot posodobitev (ang. update) in nam tako sploh ni potrebno delati varnostne kopije (ang. backup) podatkov, čeprav je v praksi to seveda zelo priporočljivo. Na tej spletni strani[4] prenesemo namestitvene pakete za izvedbo posodobitve in posledično migracije podatkov na podatkovno zbirko Percona

Server. Nato s pomočjo APT namestimo prenesene pakete. Pred tem moramo seveda zaustaviti vse storitve MySQL, ki se izvajajo. Na tej spletni strani [9] so prikazani ukazi, ki jih je potrebno vnesti, da se Percona Server pravilno namesti. Pravilno zaporedje ukazov za namestitev podatkovne zbirke Percona Server na operacijska sistema Ubuntu in Debian nam prikazuje spodnja slika (Slika 4.2).

```
$ apt-key adv --keyserver keys.gnupg.net --recv-keys 1C4CBDCDCD2EFD2A
```

To dodajte na mesto: /etc/apt/sources.list , zamenjajte 'VERSION' z vašo željeno različico.

```
deb http://repo.percona.com/apt VERSION main
deb-src http://repo.percona.com/apt VERSION main
```

Posodobite lokalni predpomnilnik.

```
$ apt-get update
```

Potem lahko namestite pakete odjemalca in strežnika.

```
# apt-get install percona-server-server-5.5 percona-server-client-5.5
```

Slika 4.2: Zaporedje ukazov pri namestitvi podatkovne zbirke Percona Server.

Če se dotaknemo še malo različic podatkovne zbirke Percona Server, lahko rečemo, da je postopek nadgradnje oz. migracije podatkov, ki smo ga prej opisali, izvedljiv v primerih, ko želimo npr. narediti migracijo podatkov iz standardne različice podatkovne zbirke MySQL 5.4 na različico Percona Server 5.4. Kadar pa želimo posodobiti tudi samo različico podatkovne zbirke npr. iz 5.4 na 5.5 pa moramo izvesti še nekaj ukazov, ki so opisani na tej spletni strani[7].

Če želimo izvesti migracijo podatkov iz podatkovne zbirke MySQL na podatkovno zbirko MariaDB ali na podatkovno zbirko Percona Server na produkcijskem strežniku je priporočljivo, zaradi prekinitve (ang. downtime), če imamo seveda to možnost, da najprej naredimo migracijo podatkov na razvijalčevem (ang. developer) strežniku.

# 5 Praktična izvedba migracije na podatkovno zbirko MariaDB

Izvedbo migracije podatkov smo izvedli tako, da smo na našem osebnem računalniku (nameščen smo imeli operacijski sistem Windows, zato praktične izvedbe migracije podatkov na podatkovno zbirko Percona Server nismo izvedli) najprej namestili podatkovno zbirko MySQL 5.6. Nato smo si iz spletne strani [6] prenesli primer podatkovne baze, ki se imenuje »classicmodels« in jo uvozili (zagnali njen skript) v našo nameščeno podatkovno zbirko MySQL. Nato smo naredili s programskim orodjem Visual Studio preprosto aplikacijo. Aplikacija ob zagonu prebere določene podatke iz podatkovne baze »classicmodels« in jih prikaže v polje »ComboBox«. Nato iz tega polja izberemo določenega kupca in pritisnemo gumb, ki sproži funkcijo »NapolniNarocilaKupca« ta pa nam napolni podatkovno tabelo z določenimi podatki, ki jih prav tako prebere iz podatkovne baze. Nato smo izvedli posodobitev podatkovne zbirke in ponovno zagnali našo testno aplikacijo in ugotovili, da aplikacija deluje, podatke ji je uspelo prebrati iz na novo nameščene podatkovne zbirke MariaDB, kar tudi pomeni, da smo uspešno migrirali podatke iz podatkovne zbirke MySQL na podatkovno zbirko MariaDB.

Naši praktični, testirani postopki migracije podatkov so podrobneje opisani v naslednjih podpoglavjih.

## 5.1 Uporabljena strojna in programska oprema

Vse teste, namestitve programske opreme, uvoz podatkovne baze, izdelavo aplikacije smo izvajali na našem osebnem računalniku z naslednjimi karakteristikami:

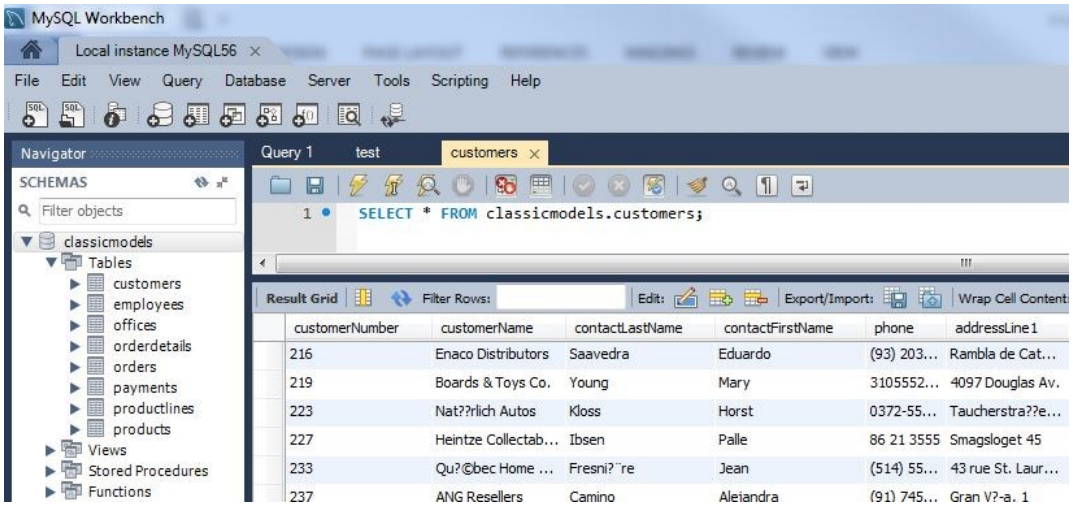
- 64 bitni operacijski sistem Windows 7 Professional (Service Pack 1)
- procesor i7 - 3632QM, 2.2 GHz
- 8 GB RAM
- Microsoft Visual Studio 2010 Professional

Značilnosti terminala so prikazane tudi na spodnji sliki (Slika 5.1).


Slika 5.1: Karakteristike terminala.

## 5.2 Izdelava testne podatkovne baze in testne aplikacije

Iz spleta smo si prenesli skript (ang. script) prosto dostopne brezplačne podatkovne baze »clasicmodels«. Nato smo zagnali orodje, imenovano MySQL Workbench, kjer smo izvršili (ang. execute) prej omenjeni skript. Izvedli smo testno poizvedbo (Slika 5.2).




Slika 5.2: Testna baza podatkov »classicmodels«.

S tem smo preverili, če je baza podatkov pravilno napolnjena s podatki. Ko smo imeli podatke pravilno nameščene v bazi smo s pomočjo programskega orodja Visual Studio naredili testni razred (ang. class) v katerega smo uvozili knjižnico »MySQL.Data«. Aplikacija vsebuje eno okno in ima polje (ComboBox_Kategorija), podatkovno tabelo (DataGridView1) in gumb (Button1). Ob zagonu aplikacije se izvede funkcija »NapolniKupce«, ki prebere podatke iz tabele »customers« in jih vrne v polje »ComboBox_Kategorija« (Slika 5.3).

```
Private Sub NapolniKupce()

    Dim sqlselect As String = ""
    sqlselect = "SELECT customername, customernumber from customers order by 1 "

    Dim conn As String = "Server=localhost;Database=classicmodels;Uid=root;"
    Dim da As New MySql.Data.MySqlClient.MySqlDataAdapter(sqlselect, conn)
    Dim ds As New Data.DataSet()
    da.Fill(ds)

    ComboBox_Kategorija.DataSource = ds.Tables(0)
    ComboBox_Kategorija.DisplayMember = "customername"
    ComboBox_Kategorija.ValueMember = "customernumber"
    ComboBox_Kategorija.SelectedIndex = 0

End Sub
```

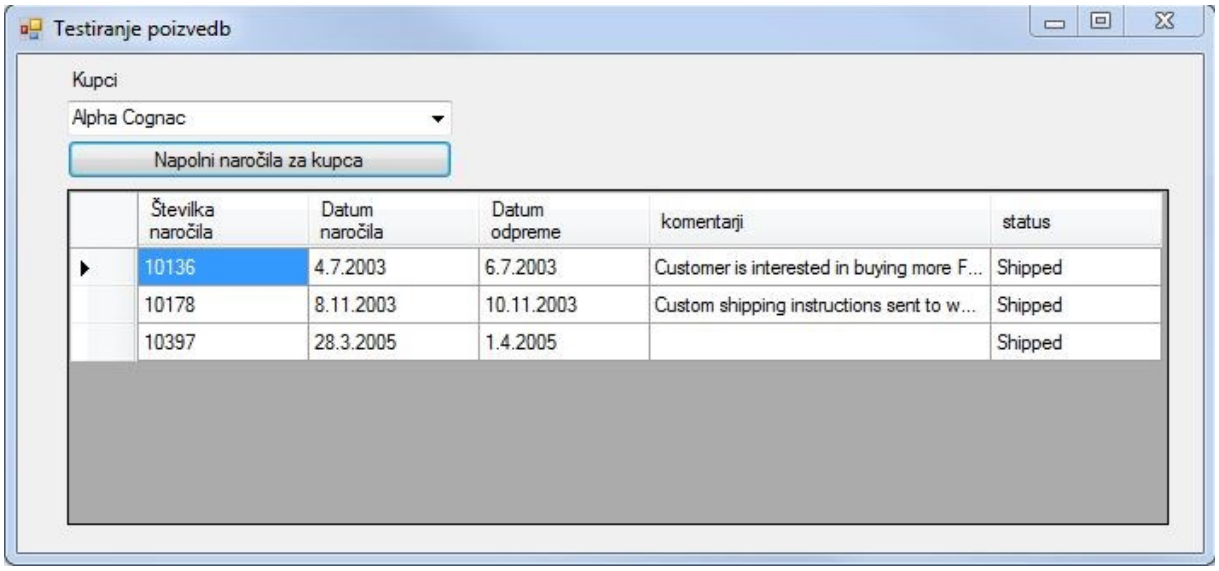
Slika 5.3: Funkcija »NapolniKupce«.

Nato iz polja »ComboBox« izberemo željenega kupca in pritisnemo gumb »Button1«, ki sproži funkcijo »NapolniNarocilaKupca« (Slika 5.4) in nam posledično napolni podatkovno tabelo »DataGridView1«.

```
    Private Sub NapolniNarocilaKupca()

        Dim sqlselect1 As String = ""
        sqlselect1 = "SELECT ordernumber, orderdate, shippeddate, comments, status  "
        sqlselect1 = sqlselect1 + "from customers, orders where
customers.customernumber=orders.customernumber  "
        sqlselect1 = sqlselect1 + "and customers.customernumber = " &
ComboBox_Kategorija.SelectedValue.ToString & " "

        Dim conn1 As String = "Server=localhost;Database=classicmodels;Uid=root;"
        Dim da1 As New MySql.Data.MySqlClient.MySqlDataAdapter(sqlselect1, conn1)
        Dim ds1 As New Data.DataSet()
        da1.Fill(ds1)

        DataGridView1.DataSource = ds1.Tables(0)

    End Sub
```

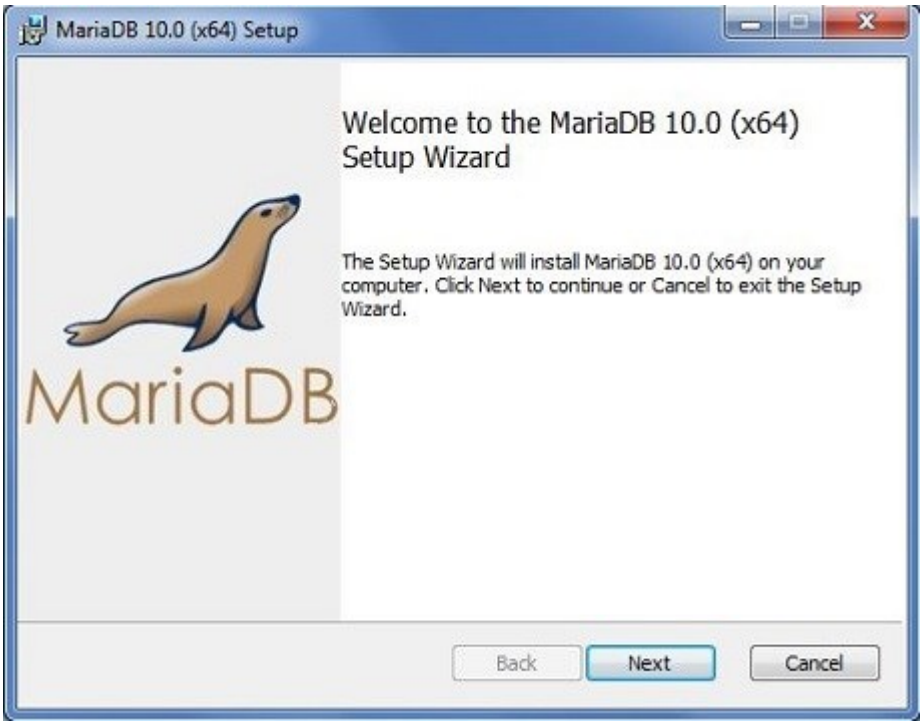
Slika 5.4: Funkcija »NapolniNarocilaKupca«.

Osnovna naloga naše aplikacije je bila ta, da se pravilno poveže na podatkovno bazo in prebere podatke, izvede poizvedbo. Zato ni bilo nobene potrebe po tem, da bi aplikacija vsebovala veliko polj, oken, gumbov, zadostovalo je že eno samo okno (Slika 5.5).



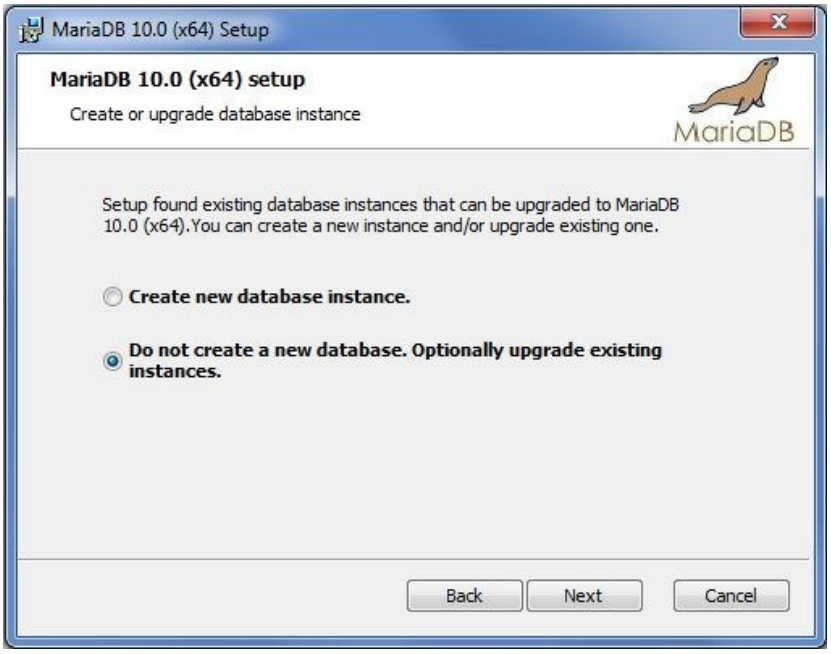

Slika 5.5: Okno testne aplikacije.

Ko je naša aplikacija opravila svojo nalogo in pravilno prebrala podatke iz podatkovne zbirke MySQL smo aplikacijo zaprli, zaustavili strežnik MySQL in pognali 64bitni MSI (Slika 5.6).



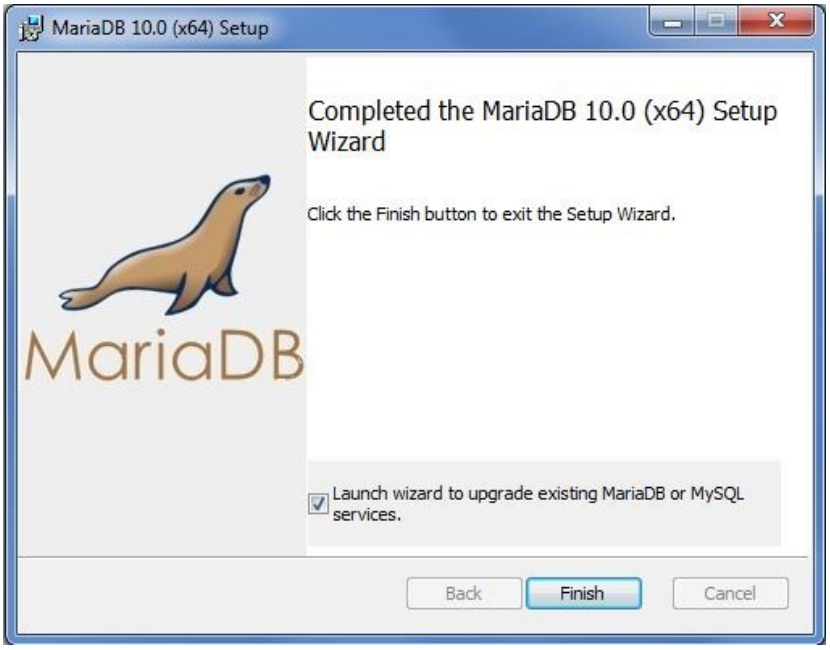

Slika 5.6: 64 bitni MSI za namestitev podatkovne zbirke MariaDB.

Pri namestitvi moramo paziti, da obkljukamo posodobi že nameščeno različico podatkovne zbirke MySQL oz. MariaDB (Slika5.7).



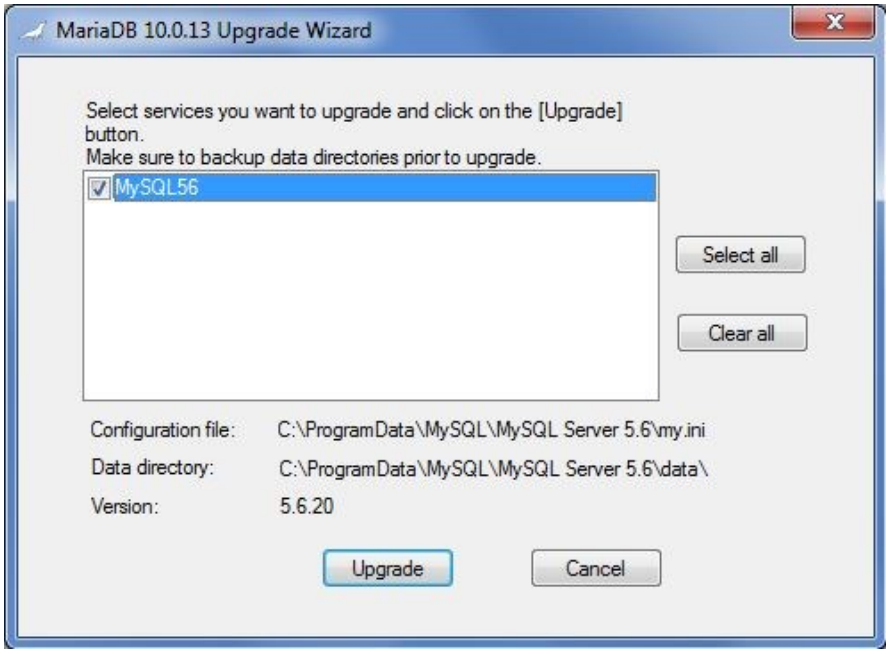

Slika 5.7: Izberemo posodobitev že obstoječega primerka podatkovne zbirke.

Po končani namestitvi ne smemo pozabiti obkljukati izbiro o zagonu GUI, orodja za posodobitev, nadgradnjo najdene že nameščene podatkovne zbirke MySQL, ki izvede ukaz »mysql_upgrade_service« (Slika 5.8).



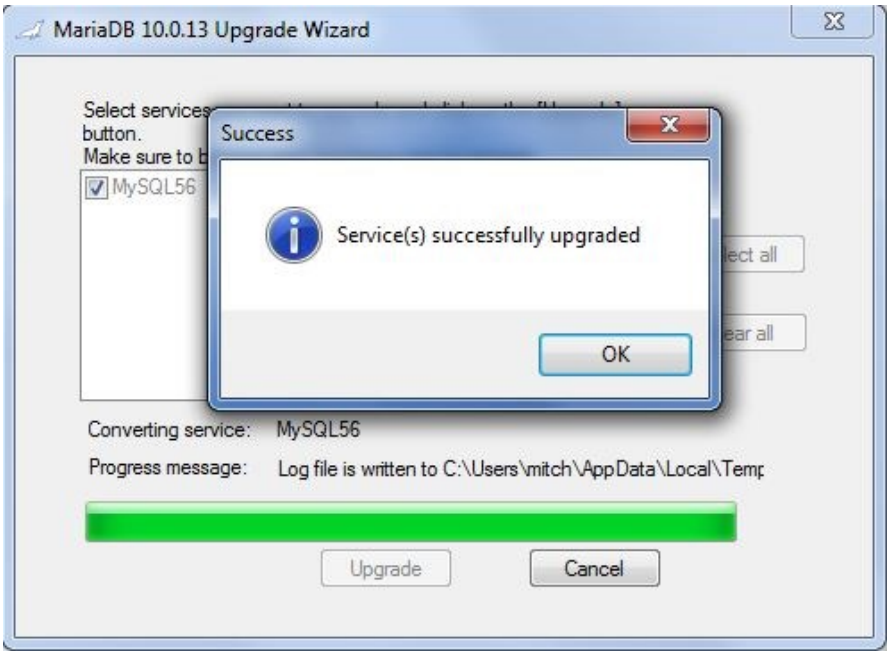

Slika 5.8: Obkljukamo zagon čarovnika za izvedbo posodobitve.

Če smo prej omenjeno izbiro res obkljukali, se nam prikaže željeni GUI, kjer obkljukamo kateri primerek že nameščene podatkovne zbirke želimo posodobiti (Slika 5.9).




Slika 5.9: GUI za posodobitev že nameščene podatkovne zbirke.

Ko prej omenjeni čarovnik za posodobitev uspešno izvrši svojo nalogo nas o tem tudi obvesti (Slika 5.10).




Slika 5.10: Obvestilo o uspešni izvedbi posodobitve.

Ko smo uspešno namestili podatkovno zbirko MariaDB smo zagnali storitve, strežnik in ponovno zagnali našo testno aplikacijo, ki je brez kakršnih koli sprememb v svoji izvorni kodi, razredu »Test«, pravilno prikazala podatke v polje »ComboBox_Kategorije« in po kliku na gumb »Button1« tudi pravilno napolnila naročila kupca v podatkovno tabelo »DataGridView1«.

## 5.3 Zaključek

Pri praktični izvedbi migracije podatkov iz podatkovne zbirke MySQL na podatkovno zbirko MariaDB smo naredili test poizvedbe iz migriranih podatkov s pomočjo aplikacije, ki smo jo naredili z programskim orodjem Visual Studio. Ugotovili smo, da razvijalcem v svojih že narejenih aplikacijah ni potrebno popravljati izvorne kode programov in da je v bistvu migracija podatkov na podatkovno zbirko MariaDB dokaj enostavna tudi za ne preveč izkušene razvijalce podatkovnih zbirk. Za to je seveda kriva predvsem binarna kompatibilnost, združljivost obravnavanih podatkovnih zbirk. Migracija podatkov na podatkovno zbirko MariaDB je zelo smotrna, smiselna saj z dokaj malo dela, dodatnega znanja in truda pridobimo bolj zmogljivo in prilagodljivo podatkovno bazo.

# 6 Sklepne ugotovitve

Ko smo raziskovali lastnosti, zmogljivosti, prednosti podatkovnih zbirk MySQL, MariaDB in Percona Server smo ugotovili, da sta slednji dve podatkovni zbirki v bistvu izboljšani različici podatkovne zbirke MySQL z dodatnimi funkcijami. Podatkovna zbirka MariaDB lahko uporablja tudi več podatkovnih strojev, vsebuje izboljšave optimizatorjev, ki omogočajo večjo zmogljivost in še bi lahko naštevali. Poleg tega je podatkovna zbirka MariaDB v lasti neprofitne organizacije, fundacije, kar je seveda tudi ena izmed velikih prednosti. Še ena velika prednost podatkovnih zbirk MariaDB in PerconaServer je, že večkrat omenjena binarna kompatibilnost z podatkovno zbirko MySQL, kar zelo olajša delo razvijalcem, ki želijo podatke migrirati. Prav tako pa se lahko njihove aplikacije kljub migraciji podatkov nemoteno uporabljajo in ni potrebnih tako rekoč nobenih posegov v njihovo izvorno kodo. To trditev smo na nek način tudi mi dokazali v petem poglavju. Naš praktični prikaz migracije podatkov je bil izveden na lokalnem strežniku (ang. localhost), na našem terminalu in migrirali smo podatke iz zelo majhne baze podatkov. Scenarij migracije podatkov na produkcijskem strežniku z zelo veliko bazo podatkov, ki jo bo izvedel Google prav gotovo ne bo tako preprosta kot v našem primeru, bodo pa zagotovo s to potezo pridobili zmoglivejšo in prilagodljivejšo bazo podatkov.

# Literatura

[1] D. Bartholomew, *Getting Started with MariaDB*, 1.poglavje. Birmingham: Packt Publishing Ltd., 2013. Dosegljivo: http://it-ebooks.info/book/3162/.

# Spletni viri

[2] Namestitveni paket *MSI package* za podatkovno zbirko MariaDB. Dosegljivo: https://downloads.mariadb.org/mariadb/10.0.13/.

[3] Namestitveni paket *MySQL Installer* za podatkovno zbirko MySQL. Dosegljivo: http://dev.mysql.com/tech-resources/articles/mysql-installer-for-windows.html.

[4] Namestitveni paket za podatkovno zbirko Percona Server. Dosegljivo: http://www.percona.com/downloads/Percona-Server-5.5/LATEST/.

[5] Percona XtraDB documentation. Dosegljivo: http://www.percona.com/doc/percona-server/5.5/?id=percona-xtradb:patch:start.

[6] Podatkovna baza »classicmodels«. Dosegljivo: http://www.mysqltutorial.org/mysql-sample-database.aspx.

[7] Posodobitev podatkovne zbirke MySQL 5.4 na različico MySQL 5.5. Dosegljivo: http://dev.mysql.com/doc/refman/5.5/en/upgrading-from-5-4.html.

[8] Postopek namestitve podatkovne zbirke MariaDB na Linux operacijske sisteme. Dosegljivo: https://downloads.mariadb.org/mariadb/repositories/#mirror=cube.

[9] Postopek namestitve podatkovne zbirke Percona Server na Linux operacijske sisteme. Dosegljivo: http://www.percona.com/doc/percona-server/5.5/installation/apt_repo.html.

[10] Prenos namestitve za podatkovno zbirko Percona Server. Dosegljivo: http://www.percona.com/downloads/Percona-Server-5.6/LATEST/binary/.

[11] The InnoDB Storage Engine, Chapter 14. Dosegljivo: http://dev.mysql.com/doc/refman/5.7/en/innodb-storage-engine.html.

# Priloge

## ➢ Priloga 1

Skripta naše testne podatkovne baze »classicmodels«, ki je prosto dostopna na spletu [6].

```
CREATE DATABASE `classicmodels` ;

USE `classicmodels`;

/*Table structure for table `customers` */

DROP TABLE IF EXISTS `customers`;

CREATE TABLE `customers` (
  `customerNumber` int(11) NOT NULL,
  `customerName` varchar(50) NOT NULL,
  `contactLastName` varchar(50) NOT NULL,
  `contactFirstName` varchar(50) NOT NULL,
  `phone` varchar(50) NOT NULL,
  `addressLine1` varchar(50) NOT NULL,
  `addressLine2` varchar(50) DEFAULT NULL,
  `city` varchar(50) NOT NULL,
  `state` varchar(50) DEFAULT NULL,
  `postalCode` varchar(15) DEFAULT NULL,
  `country` varchar(50) NOT NULL,
  `salesRepEmployeeNumber` int(11) DEFAULT NULL,
  `creditLimit` double DEFAULT NULL,
  PRIMARY KEY (`customerNumber`),
  KEY `salesRepEmployeeNumber` (`salesRepEmployeeNumber`),
  CONSTRAINT `customers_ibfk_1` FOREIGN KEY (`salesRepEmployeeNumber`) REFERENCES
`employees` (`employeeNumber`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

/*Data for the table `customers` */

LOCK TABLES `customers` WRITE;

insert  into
`customers`(`customerNumber`,`customerName`,`contactLastName`,`contactFirstName`,`phon
e`,`addressLine1`,`addressLine2`,`city`,`state`,`postalCode`,`country`,`salesRepEmploy
eeNumber`,`creditLimit`)
values
(103,'Atelier graphique','Schmitt','Carine ','40.32.2555','54, rue
Royale',NULL,'Nantes',NULL,'44000','France',1370,21000),
(112,'Signal Gift Stores','King','Jean','7025551838','8489 Strong St.',NULL,'Las
Vegas','NV','83030','USA',1166,71800),
(114,'Australian Collectors, Co.','Ferguson','Peter','03 9520 4555','636 St Kilda
Road','Level 3','Melbourne','Victoria','3004','Australia',1611,117300),
(119,'La Rochelle Gifts','Labrune','Janine ','40.67.8555','67, rue des Cinquante
Otages',NULL,'Nantes',NULL,'44000','France',1370,118200),
(121,'Baane Mini Imports','Bergulfsen','Jonas ','07-98 9555','Erling Skakkes gate
78',NULL,'Stavern',NULL,'4110','Norway',1504,81700);
UNLOCK TABLES;
```

```
/*Table structure for table `employees` */

DROP TABLE IF EXISTS `employees`;

CREATE TABLE `employees` (
  `employeeNumber` int(11) NOT NULL,
  `lastName` varchar(50) NOT NULL,
  `firstName` varchar(50) NOT NULL,
  `extension` varchar(10) NOT NULL,
  `email` varchar(100) NOT NULL,
  `officeCode` varchar(10) NOT NULL,
  `reportsTo` int(11) DEFAULT NULL,
  `jobTitle` varchar(50) NOT NULL,
  PRIMARY KEY (`employeeNumber`),
  KEY `reportsTo` (`reportsTo`),
  KEY `officeCode` (`officeCode`),
  CONSTRAINT `employees_ibfk_2` FOREIGN KEY (`officeCode`) REFERENCES `offices`
(`officeCode`),
  CONSTRAINT `employees_ibfk_1` FOREIGN KEY (`reportsTo`) REFERENCES `employees`
(`employeeNumber`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

/*Data for the table `employees` */

LOCK TABLES `employees` WRITE;

insert  into
`employees`(`employeeNumber`,`lastName`,`firstName`,`extension`,`email`,`officeCode`,`
reportsTo`,`jobTitle`)
values
(1002,'Murphy','Diane','x5800','dmurphy@classicmodelcars.com','1',NULL,'President'),
(1056,'Patterson','Mary','x4611','mpatterso@classicmodelcars.com','1',1002,'VP Sale'),
(1076,'Firrelli','Jeff','x9273','jfirrelli@classicmodelcars.com','1',1002,'VP Mark'),
(1088,'Patterson','William','x4871','wpatterson@classicmodelcars.com','6',1056,'Sal'),
(1102,'Bondur','Gerard','x5408','gbondur@classicmodelcars.com','4',1056,'Sal(EMEA)');
UNLOCK TABLES;

/*Table structure for table `offices` */

DROP TABLE IF EXISTS `offices`;

CREATE TABLE `offices` (
  `officeCode` varchar(10) NOT NULL,
  `city` varchar(50) NOT NULL,
  `phone` varchar(50) NOT NULL,
  `addressLine1` varchar(50) NOT NULL,
  `addressLine2` varchar(50) DEFAULT NULL,
  `state` varchar(50) DEFAULT NULL,
  `country` varchar(50) NOT NULL,
  `postalCode` varchar(15) NOT NULL,
  `territory` varchar(10) NOT NULL,
  PRIMARY KEY (`officeCode`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;


/*Data for the table `offices` */

LOCK TABLES `offices` WRITE;
```

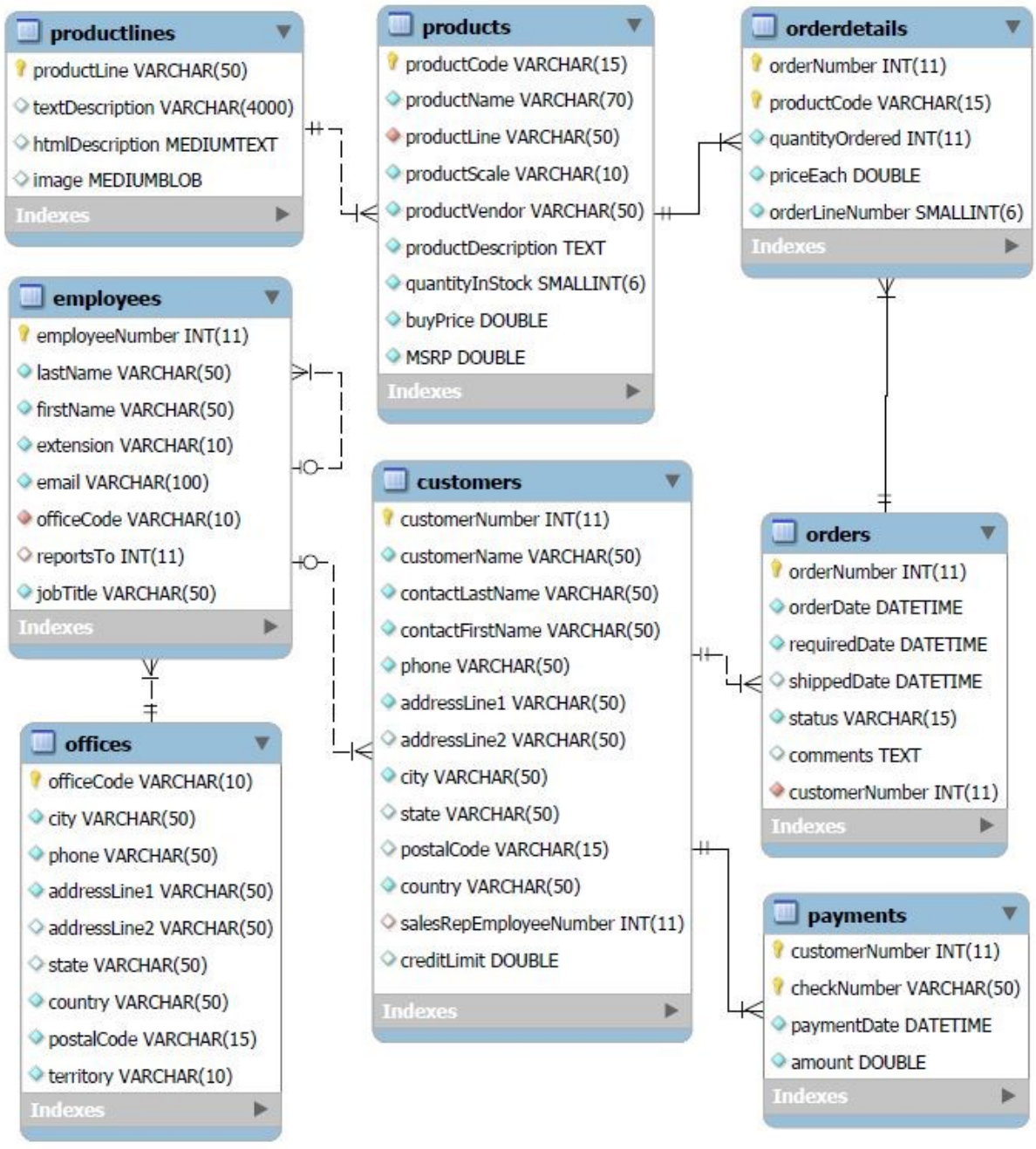
```
insert  into
`offices`(`officeCode`,`city`,`phone`,`addressLine1`,`addressLine2`,`state`,`country`,
`postalCode`,`territory`)
values
('1','San Francisco','+1 650 219 4782','100 Market Street','Suite
300','CA','USA','94080','NA'),
('2','Boston','+1 215 837 0825','1550 Court Place','Suite
102','MA','USA','02107','NA'),
('3','NYC','+1 212 555 3000','523 East 53rd Street','apt. 5A','NY',
'USA','10022','NA'),
('4','Paris','+33 14 723 4404','abbans',NULL,NULL,'France','75017','EMEA'),
('5','Tokyo','+81 33 224 5000','4-1 Kioicho',NULL,'Chiyoda-Ku','Japan','102-
8578','Japan'));

UNLOCK TABLES;

/*Table structure for table `orderdetails` */

DROP TABLE IF EXISTS `orderdetails`;

CREATE TABLE `orderdetails` (
  `orderNumber` int(11) NOT NULL,
  `productCode` varchar(15) NOT NULL,
  `quantityOrdered` int(11) NOT NULL,
  `priceEach` double NOT NULL,
  `orderLineNumber` smallint(6) NOT NULL,
  PRIMARY KEY (`orderNumber`,`productCode`),
  KEY `productCode` (`productCode`),
  CONSTRAINT `orderdetails_ibfk_2` FOREIGN KEY (`productCode`) REFERENCES `products`
(`productCode`),
  CONSTRAINT `orderdetails_ibfk_1` FOREIGN KEY (`orderNumber`) REFERENCES `orders`
(`orderNumber`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

/*Data for the table `orderdetails` */

LOCK TABLES `orderdetails` WRITE;

insert  into
`orderdetails`(`orderNumber`,`productCode`,`quantityOrdered`,`priceEach`,`orderLineNum
ber`)
values
(10100,'S18_1749',30,136,3),
(10100,'S18_2248',50,55.09,2),
(10100,'S18_4409',22,75.46,4),
(10100,'S24_3969',49,35.29,1),
(10101,'S18_2325',25,108.06,4);
UNLOCK TABLES;

/*Table structure for table `orders` */

DROP TABLE IF EXISTS `orders`;

CREATE TABLE `orders` (
  `orderNumber` int(11) NOT NULL,
  `orderDate` date NOT NULL,
  `requiredDate` date NOT NULL,
  `shippedDate` date DEFAULT NULL,
  `status` varchar(15) NOT NULL,
  `comments` text,
```

```
  `customerNumber` int(11) NOT NULL,
  PRIMARY KEY (`orderNumber`),
  KEY `customerNumber` (`customerNumber`),
  CONSTRAINT `orders_ibfk_1` FOREIGN KEY (`customerNumber`) REFERENCES `customers`
(`customerNumber`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

/*Data for the table `orders` */

LOCK TABLES `orders` WRITE;

insert  into
`orders`(`orderNumber`,`orderDate`,`requiredDate`,`shippedDate`,`status`,`comments`,`c
ustomerNumber`)
values
(10100,'2003-01-06','2003-01-13','2003-01-10','Shipped',NULL,363),
(10101,'2003-01-09','2003-01-18','2003-01-11','Shipped','Check on availability.',128),
(10102,'2003-01-10','2003-01-18','2003-01-14','Shipped',NULL,181),
(10103,'2003-01-29','2003-02-07','2003-02-02','Shipped',NULL,121),
(10104,'2003-01-31','2003-02-09','2003-02-01','Shipped',NULL,141),
(10105,'2003-02-11','2003-02-21','2003-02-12','Shipped',NULL,145))
UNLOCK TABLES;

/*Table structure for table `payments` */

DROP TABLE IF EXISTS `payments`;

CREATE TABLE `payments` (
  `customerNumber` int(11) NOT NULL,
  `checkNumber` varchar(50) NOT NULL,
  `paymentDate` date NOT NULL,
  `amount` double NOT NULL,
  PRIMARY KEY (`customerNumber`,`checkNumber`),
  CONSTRAINT `payments_ibfk_1` FOREIGN KEY (`customerNumber`) REFERENCES `customers`
(`customerNumber`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

/*Data for the table `payments` */

LOCK TABLES `payments` WRITE;

insert  into `payments`(`customerNumber`,`checkNumber`,`paymentDate`,`amount`)
values
(103,'HQ336336','2004-10-19',6066.78),
(103,'JM555205','2003-06-05',14571.44),
(103,'OM314933','2004-12-18',1676.14),
(112,'BO864823','2004-12-17',14191.12),
(112,'HQ55022','2003-06-06',32641.98);
UNLOCK TABLES;



/*Table structure for table `productlines` */

DROP TABLE IF EXISTS `productlines`;

CREATE TABLE `productlines` (
  `productLine` varchar(50) NOT NULL,
  `textDescription` varchar(4000) DEFAULT NULL,
  `htmlDescription` mediumtext,
```

```
  `image` mediumblob,
  PRIMARY KEY (`productLine`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

/*Data for the table `productlines` */

LOCK TABLES `productlines` WRITE;

insert  into `productlines`(`productLine`,`textDescription`,`htmlDescription`,`image`)
values
('Classic Cars','Attention car enthusiast....',NULL,NULL),
('Motorcycles','Our motorcycles are state of the art replicas...',NULL,NULL),
('Planes','Unique, diecast airplane and helicopter...',NULL,NULL),
('Ships','The perfect holiday or anniversary gift for executives...',NULL,NULL),
('Trucks and Buses','The Truck and Bus models....',NULL,NULL);

UNLOCK TABLES;

/*Table structure for table `products` */

DROP TABLE IF EXISTS `products`;

CREATE TABLE `products` (
  `productCode` varchar(15) NOT NULL,
  `productName` varchar(70) NOT NULL,
  `productLine` varchar(50) NOT NULL,
  `productScale` varchar(10) NOT NULL,
  `productVendor` varchar(50) NOT NULL,
  `productDescription` text NOT NULL,
  `quantityInStock` smallint(6) NOT NULL,
  `buyPrice` double NOT NULL,
  `MSRP` double NOT NULL,
  PRIMARY KEY (`productCode`),
  KEY `productLine` (`productLine`),
  CONSTRAINT `products_ibfk_1` FOREIGN KEY (`productLine`) REFERENCES `productlines`
(`productLine`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

/*Data for the table `products` */

LOCK TABLES `products` WRITE;

insert  into
`products`(`productCode`,`productName`,`productLine`,`productScale`,`productVendor`,`p
roductDescription`,`quantityInStock`,`buyPrice`,`MSRP`) values
('S10_1678','1969 Harley Davidson Ultimate Chopper','Motorcycles','1:10','Min Lin
Diecast','This  attention.....',7933,48.81,95.7),
('S10_1949','1952 Alpine Renault 1300','Classic Cars','1:10','Classic Metal
Creations','Turnable front wheels...',7305,98.58,214.3),
('S10_2016','1996 Moto Guzzi 1100i','Motorcycles','1:10','Highway 66 Mini
Classics','Official Moto Guzzi logo...',6625,68.99,118.94);
UNLOCK TABLES;
```

## ➢ Priloga 2

Konceptualni model naše testne baze »classicmodels«, ki smo jo uporabljali za testiranje.



## ➢ Priloga 3

Razred »Test«, ki smo ga naredili s programskim orodjem Visual Studio 2010. Z njim smo izvedli testne poizvedbe iz podatkovne baze MySQL in kasneje iz MariaDB.

```
Imports MySql.Data
Imports System.Data
Imports System.Configuration
Imports System.Data.SqlClient
Imports System.Data.OleDb

Public Class Test

    Private Sub Form1_Load(sender As System.Object, e As System.EventArgs) Handles
MyBase.Load
        NapolniKupce()
    End Sub

    Private Sub NapolniKupce()

        Dim sqlselect As String = ""
        sqlselect = "SELECT customername, customernumber from customers order by 1 "

        Dim conn As String = "Server=localhost;Database=classicmodels;Uid=root;"
        Dim da As New MySql.Data.MySqlClient.MySqlDataAdapter(sqlselect, conn)
        Dim ds As New Data.DataSet()
        da.Fill(ds)

        ComboBox_Kategorija.DataSource = ds.Tables(0)
        ComboBox_Kategorija.DisplayMember = "customername"
        ComboBox_Kategorija.ValueMember = "customernumber"
        ComboBox_Kategorija.SelectedIndex = 0

    End Sub

    Private Sub NapolniNarocilaKupca()

        Dim sqlselect1 As String = ""
        sqlselect1 = "SELECT ordernumber, orderdate, shippeddate, comments, status  "
        sqlselect1 = sqlselect1 + "from customers, orders where
customers.customernumber=orders.customernumber  "
        sqlselect1 = sqlselect1 + "and customers.customernumber = " &
ComboBox_Kategorija.SelectedValue.ToString & " "

        Dim conn1 As String = "Server=localhost;Database=classicmodels;Uid=root;"
        Dim da1 As New MySql.Data.MySqlClient.MySqlDataAdapter(sqlselect1, conn1)
        Dim ds1 As New Data.DataSet()
        da1.Fill(ds1)

        DataGridView1.DataSource = ds1.Tables(0)

    End Sub

    Private Sub Button1_Click(sender As System.Object, e As System.EventArgs) Handles
Button1.Click
        NapolniNarocilaKupca()
    End Sub
End Class
```